INJECT A TURN, NOT A TOKEN: LEARNING WHERE TO REDIRECT LLM REASONING FOR SELF-CORRECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models struggle to reliably identify and correct errors in their own reasoning. Unlike prior training-based approaches that fine-tune an LLM to generate revisions or feedback, we learn *where* to intervene in failed rollouts and determine *how* to redirect them. To choose where to intervene, we train a 130K-parameter MLP head on a frozen encoder using only final-answer correctness. For redirection, we find that under a fixed generation budget, the tested inline injections fail to reliably redirect reasoning, whereas presenting the same instruction in a new user turn changes the model's course. The resulting correction framework, UPTURN (User-Prompt TURN injection at a learned position), requires no LLM fine-tuning: it retains a selected prefix, injects an instruction as a new user turn, and regenerates the suffix. Across four reasoners and six math benchmarks, UPTURN improves average accuracy in every setting over a 32-rollout baseline. It outperforms self-correction and reflection-token injection baselines under matched correction budgets, transfers across reasoners and to GPQA-Diamond without retraining, and complements GRPO and self-correction methods.

1 INTRODUCTION

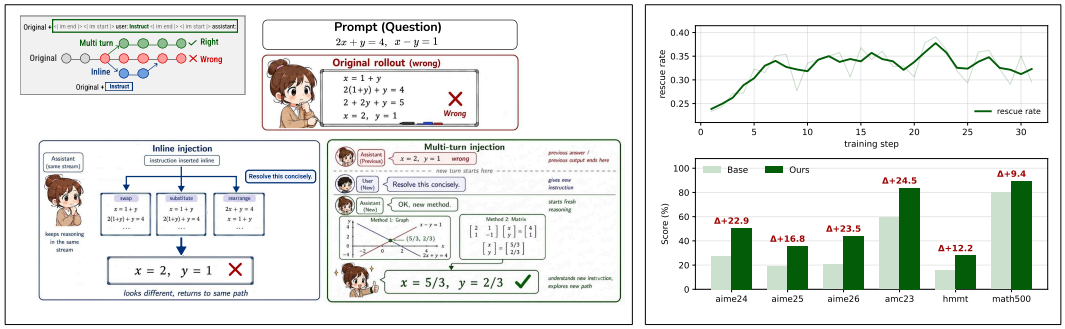


Figure 1: **Structure, not content alone, enables redirection. (Left)** The same instruction is absorbed inline but redirects reasoning in a new user turn. **(Right)** Learning where to intervene raises rescue rate (the fraction of failed rollouts corrected; top) and six-benchmark mean accuracy (bottom).

Large language models (LLMs) reason *autoregressively*, so a single mistake can derail subsequent reasoning, making effective correction important. Yet, given only its own rollout, a model may struggle to determine whether it has erred, and intrinsic self-correction without external feedback can even degrade reasoning accuracy (Huang et al., 2024). This motivates learning a correction signal from data. Existing methods do so by fine-tuning the reasoner to produce a revised answer (Kumar et al., 2025; Qu et al., 2024) or by training an external critic or verifier to provide step-level feedback or scores (Lightman et al., 2024; Paul et al., 2024). Although effective, these methods require substantial correction data and typically optimize an LLM-scale component.

This raises a question: "Does correction really require training an entire LLM?" Repeated sampling shows that LLMs can solve far more problems than in a single attempt, with pass@k increasing as k grows (Brown et al., 2024). Yet individual rollouts can still fail: because autoregressive generation conditions each step on the preceding reasoning, even a single mistake can steer the subsequent

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.

trajectory away from a correct solution. A failed rollout need not involve many independent errors. Correction may therefore involve redirecting the model toward a solution it can already reach, rather than teaching it to reason *de novo*. We thus frame correction around two questions: *where* to intervene in a failed rollout and *how* to redirect it.

We study these questions in a *fixed-budget prefix-regeneration* setting: after selecting a cut, we retain the prefix and regenerate only the suffix under the original response-token limit. Answering the *where* question requires selecting a single cut position in a failed rollout rather than generating a revised answer or written feedback. We train a 130K-parameter MLP head on a frozen Qwen3-0.6B encoder to favor cut positions that more often lead to correct answers upon regeneration. To address *how* to redirect the rollout, we consider token injection. Reflection tokens such as "`wait`" mark high-entropy forks where a reasoning path can branch in new directions (Wang et al., 2025), suggesting that they might steer a rollout. Indeed, s1 (Muennighoff et al., 2025) improves accuracy by appending "`wait`" when the model attempts to stop, extending its reasoning. This suggests a simple recipe: learn where to intervene, inject a reflection token, and let the model continue.

However, our experiments reveal a critical limitation of inline injection under the fixed-budget protocol. Across the tested reflection tokens ("`wait`", "`hmm`"), explicit instructions ("`Solve with a different way.`"), and even a task-irrelevant word ("`Turtle`"), inline injection fails to reliably redirect the rollout. The key difference is conversational *structure*: the same instruction can redirect reasoning when delivered as a new user turn (Figure 1, left), whereas inline delivery does not reliably do so. These results suggest that the model responds more effectively to an instruction delivered in a user turn than to the same text inserted into its own continuation. We refer to this structural transition from an assistant continuation to a new user turn as a *turn boundary*.

Putting these two pieces together, we propose UPTURN (User-Prompt TURN injection at a learned position), a correction framework that requires no LLM fine-tuning. UPTURN keeps the reasoner and the 0.6B encoder frozen and trains a position policy, implemented as a 130K-parameter MLP head, to select where a fixed instruction is injected through a turn boundary. A separately trained 66K-parameter gate identifies which rollouts to correct at inference. Rather than identify the error or generate a fix, the position policy chooses where to redirect the rollout, while the frozen reasoner performs the correction itself. Averaged across four reasoners and six math benchmarks, UPTURN raises accuracy from 37.1% to 55.3% (Figure 1, right). It outperforms self-correction and reflection-token injection baselines under matched correction budgets, as well as heuristic cuts and inline injection. Our contributions are threefold:

- We show that conversational structure matters for correction: the tested inline injections fail to reliably redirect reasoning, whereas the same instruction delivered through a turn boundary can.
- We learn cut positions with a 130K-parameter head on a frozen encoder, using only regenerated rollouts' final-answer correctness, without step-level annotations or reasoner fine-tuning.
- We demonstrate consistent accuracy gains across four reasoners and six math benchmarks, transfer across reasoners and to GPQA-Diamond without retraining, and complementary gains on top of GRPO and existing self-correction methods.

2 Related Work

Prompting-based self-correction. Prompting-based methods improve outputs without updating model weights, typically by having the model critique or reflect on its own attempt and then revise it (Madaan et al., 2023). Reflexion (Shinn et al., 2023), for example, generates a verbal reflection on the previous attempt and uses it to guide a fresh attempt. These methods rely on the reasoner to diagnose its own errors, which can be unreliable for reasoning (Huang et al., 2024; Tie et al., 2025). UPTURN asks the reasoner for neither a critique nor a rewrite: a learned position policy selects where to intervene, and a fixed user-turn instruction redirects the existing rollout.

Training-based self-correction. Training-based methods learn corrective signals from data, some by fine-tuning the reasoner to revise its own answers (Qu et al., 2024); SCoRe (Kumar et al., 2025), for example, trains the model with RL to produce a revised second attempt conditioned on its first. Others train a separate model to localize errors, such as a critic whose step-level feedback guides the reasoner's revision (Paul et al., 2024) or a process reward model that scores each step (Lightman

et al., 2024). These approaches typically train an LLM-scale component using correction data, such as revision pairs or step-level annotations. UPTURN instead trains lightweight heads to select which rollouts to correct and *where* to intervene, rather than generating revisions, critiques, or step-level scores, while keeping the reasoner frozen.

Reflection-token injection. Several methods improve inference-time performance by injecting or promoting reflection tokens in a frozen model's reasoning. They append a token such as "`wait`" when the model attempts to stop (Muennighoff et al., 2025; Ringel et al., 2025), or encourage reflection tokens during generation on a predetermined schedule through direct insertion or logit upweighting (Zhang et al., 2025; Fan et al., 2026). These methods steer an ongoing generation from within the model's own response, whereas we study post-hoc correction of completed rollouts flagged as likely incorrect. In this setting, we find that how the text is delivered matters: under a matched generation budget, the tested inline injections fail to reliably redirect reasoning even at learned cut positions (§6.5), whereas the same instruction delivered as a separate user turn does.

3 Preliminaries

Autoregressive generation over chat turns. An autoregressive language model p_θ draws each token from the next-token distribution conditioned on all preceding tokens. We represent a conversation as alternating single-message user and assistant turns, written $[\textsc{user}: q]$ and $[\textsc{asst}: o]$ and concatenated with $\|$. Thus, a question q and a response o form the two-turn conversation $[\textsc{user}: q] \,\|\, [\textsc{asst}: o]$. Before generation, the chat template tokenizes the conversation and wraps each turn in special tokens marking where the turn begins and ends (e.g., <|im_start|>user ... <|im_end|> for Qwen (Yang et al., 2025)). The response is then generated token by token as

$$o_t \sim p_\theta\big(\cdot \mid [\textsc{user}: q] \,\|\, [\textsc{asst}: o_{<t}]\big), \qquad t = 1, \ldots, T, \tag{1}$$

where $o_{<t} = (o_1, \ldots, o_{t-1})$, T is the response length, and the assistant turn remains open.

Mathematical reasoning with LLMs. In the standard math setup, the model answers each problem with a chain-of-thought rollout o (Wei et al., 2022) in a single assistant turn via Eq. (1), with no further turns. The answer is the content of the last `\boxed{}` expression in the rollout, and the reward is its exact match with the ground truth $a^\star$,

$$R(q, o) \;=\; \mathbf{1}[\texttt{answer}(o) = a^\star] \;\in\; \{0, 1\}. \tag{2}$$

Group Relative Policy Optimization. GRPO (Shao et al., 2024) is a policy-gradient method widely used to post-train LLM reasoners. Unlike PPO (Schulman et al., 2017), it removes the learned value network: for each input, it samples G responses and forms advantages by centering and scaling their rewards with the group mean and standard deviation. It then applies PPO's clipped surrogate objective, typically with a KL penalty to a reference policy. We adapt this group-relative construction to train the position policy, with the modifications described in Sec. 5.2.

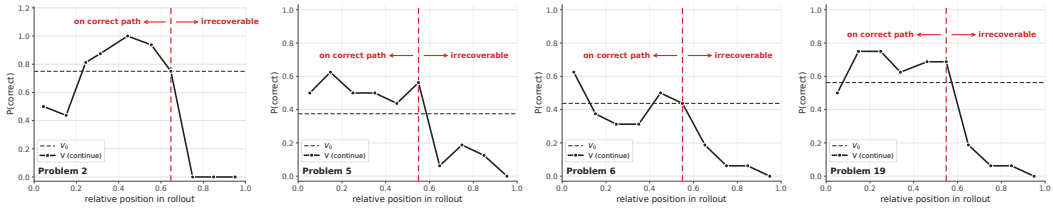
4 Motivation

Reinforcement Learning with Verifiable Rewards (RLVR) (Shao et al., 2024; Liu et al., 2025) has enabled LLMs to produce high-quality mathematical solutions. Prior work (Wang et al., 2025) shows that RLVR updates concentrate on high-entropy "forking" tokens (e.g., `Wait`, `Alternatively`), with smaller updates at the more numerous low-entropy tokens. These high-entropy tokens correspond to decision points where the reasoning can branch.

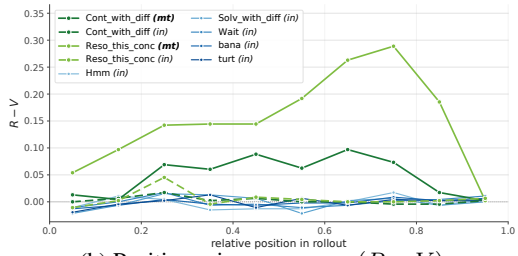
This observation motivates the following question for reasoning correction: If RLVR updates concentrate on such decision points during training, could intervening at analogous points during inference help correct a failed reasoning trace? A natural candidate is token injection: select a branching point and inject a reflection token to encourage a different continuation without retraining the reasoner.

4.1 Recovery Depends on the Retained Prefix

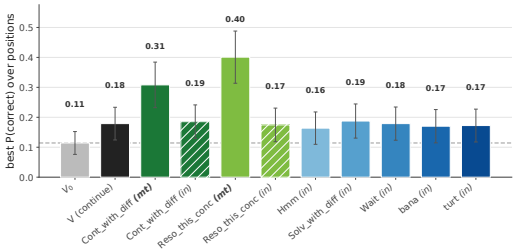
Failed traces may contain useful prefixes that can lead to correct answers. Restarting from scratch discards these prefixes, whereas regeneration from a suitable cut preserves them.



(a) Prefix-dependent recovery



(b) Position-wise rescue gap $(R - V)$



(c) Oracle best rescue rate

Figure 2: **Why turn-boundary injection is needed.** **(a)** Prefix value along failed rollouts, computed as the mean success rate over 16 continuations sampled from each prefix. **(b)** Position-wise rescue gap $R-V$ for inline (in) and multi-turn (mt) injection, where R is post-injection success and V is prefix value. **(c)** Oracle best rescue rate obtained by selecting, for each problem and phrase, the position with highest post-injection success.

To test this, we collected failed Qwen3-4B (Thinking on) traces on AIME 2026 and extracted ten prefixes at equally spaced token positions from each trace. We sample 16 such continuations per prefix without any injected text and define the continuation success rate V of each prefix as the fraction of continuations with correct final answers. As a reference, V_0 denotes the success rate when the model generates from the question alone, with no retained prefix. In the traces shown in Figure 2a, continuation success rates are high at earlier prefixes but drop sharply and remain low at later ones. This suggests that the choice of retained prefix can substantially affect the model's ability to recover.

The sharp decline may mark a branching point beyond which recovery becomes difficult. This motivates cutting before the transition to preserve a prefix from which the model can still recover.

4.2 Inline vs. Multi-turn Injection

The previous subsection motivates *where* to intervene: before a potential branching point marked by a sharp drop in continuation success. We now ask *how* to redirect reasoning at a given cut position.

Using the recovery protocol in Sec. 4.1, we compare injection variants at the sampled cut positions. At each position t, we retain the prefix $o_{\leq t}$, discard the remaining suffix, and append the injected text c within the same assistant turn before regeneration. We test reflection tokens such as `Wait` and `Hmm`, unrelated words such as `turtle` and `banana`, and longer instructions such as "`Solve with a different way.`". For each rollout and cut position, only c varies across the inline variants. Inline injection is expressed as

$$\mathcal{C}_{\text{in}} = [\text{USER}: q] \,\|\, [\text{ASST}: o_{\leq t},\ c], \tag{3}$$

where $c = (c_1, \ldots, c_m)$ denotes the injected text. Across the tested texts, inline injection yields similar continuation success rates. Surprisingly, these rates remain close to those obtained by regenerating from the prefix without injection (Figures 2b and 2c).

We next deliver the same text c through the chat template as a new user turn. We close the current assistant turn, insert a user turn containing c, and open a new assistant turn for regeneration:

$$\mathcal{C}_{\text{mt}} = [\text{USER}: q] \,\|\, [\text{ASST}: o_{\leq t}] \,\|\, [\text{USER}: c] \,\|\, [\text{ASST}: \cdot]. \tag{4}$$

This multi-turn format yields substantially higher continuation success rates than inline injection.

These results show that the injection format matters beyond instruction wording. Both contexts use the same frozen model p_θ and injected text c, but differ in their conditioning format. In $\mathcal{C}_{\text{in}}$, the short c ($m \ll t$) follows a long sequence of assistant tokens and may therefore be interpreted as part of the

ongoing reasoning rather than as a revision signal. In $\mathcal{C}_{\text{mt}}$, the chat template instead presents the same c as an *external* instruction in a new user turn, which may help redirect the continuation. Figure 6 in Appendix A illustrates this contrast: inline continuation stays on the original path, whereas a new user turn redirects reasoning.

5 METHOD

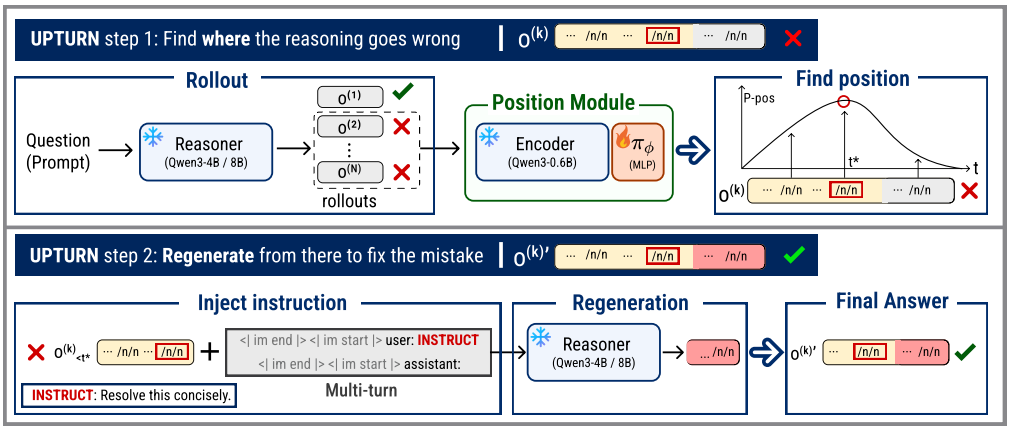


Figure 3: **Overview of UPTURN. (Step 1)** A frozen 0.6B encoder and MLP head f_ϕ score paragraph boundaries (`\n\n`) in failed rollouts (✗); the policy selects a cut $t^\star$. **(Step 2)** UPTURN retains the prefix, inserts a fixed instruction as a new user turn, and regenerates in a new assistant turn, recovering a correct answer (✓).

Motivated by §4, we correct a failed rollout by discarding the suffix after a selected position, inserting a fixed instruction c as a new user turn, and letting the frozen reasoner regenerate (Figure 3). The reasoner p_θ (e.g., Qwen3-4B) produces both the original and revised rollouts, while the position policy π_ϕ selects the cut. All LLM parameters and the injection instruction remain fixed; only lightweight heads are trained.

5.1 POSITION POLICY

Given a question q and a failed rollout o, a frozen Qwen3-0.6B causal encoder E produces a hidden state h_t at each token o_t. Because the encoder is causal, each candidate's score $f_\phi(h_t)$ depends only on the question and retained prefix $o_{\le t}$, not on the discarded suffix. To reduce the cost of evaluating regeneration rewards at every token, we restrict cuts to paragraph boundaries (`\n\n`), natural breaks between reasoning steps. Let $\mathcal{T}(o)$ denote the token positions at the ends of these boundaries. Appendix H shows that next-token entropy is elevated at paragraph boundaries, supporting these candidate cuts. Cutting at t retains $o_{\le t}$ and discards $o_{>t}$. We train only a two-layer MLP head f_ϕ (130K parameters) and normalize its scores over candidates to obtain the policy:

$$\pi_\phi(t \mid q, o) = \frac{\exp\big(f_\phi(h_t)\big)}{\sum_{t' \in \mathcal{T}(o)} \exp\big(f_\phi(h_{t'})\big)}. \tag{5}$$

5.2 TRAINING THE POSITION POLICY

Collecting failed rollouts. We sample N rollouts per prompt and train on all failures ($R(q, o) = 0$). Since all-success prompts offer nothing to correct, we sample additional prompt batches until obtaining B prompts with at least one failure, similar to dynamic sampling (Yu et al., 2025).

Multi-turn instruction injection. Based on Sec. 4.2, given a failed rollout and a cut position t, we close the assistant turn after the prefix $o_{\le t}$, insert the fixed instruction c ("`Resolve this concisely.`") as a new user turn, and open a new assistant turn,

$$\mathrm{ctx}(q, o_{\le t}, c) = [\textsc{user}: q] \,\|\, [\textsc{asst}: o_{\le t}] \,\|\, [\textsc{user}: c] \,\|\, [\textsc{asst}: \cdot], \tag{6}$$

Here, $[\textsc{asst}: \cdot]$ denotes the new assistant turn from which generation begins. We use this injection format during both training and inference. The reasoner generates at most $L_{\max} - t$ suffix tokens, keeping the retained prefix and regenerated suffix within the original response-token budget $L_{\max}$.

Reward and objective. We reward cuts by regeneration success. For each failed rollout, we sample G cuts $\{t^{(1)}, \ldots, t^{(G)}\}$ without replacement from $\pi_\phi(\cdot \mid q, o)$ as one GRPO group. At each $t^{(i)}$, the reasoner generates M suffixes $\tilde{o}^{(i,s)} \sim p_\theta(\cdot \mid \mathrm{ctx}(q, o_{\leq t^{(i)}}, c))$. We score each cut by averaging the final-answer correctness of its M regenerated suffixes:

$$r_i = \frac{1}{M} \sum_{s=1}^{M} R\big(q, \tilde{o}^{(i,s)}\big). \tag{7}$$

With p_θ and c fixed, averaging correctness over M regenerated suffixes reduces reward-estimation variance. Unlike standard GRPO, which groups independent responses, we group cuts from one failed rollout to compare positions while holding the trajectory fixed. Following Dr.GRPO (Liu et al., 2025), we drop standard-deviation normalization and use the mean-subtracted advantage $\hat{A}_i = r_i - \frac{1}{G}\sum_{j=1}^{G} r_j$, with importance ratio $\rho_i = \pi_\phi(t^{(i)} \mid q, o) / \pi_{\phi_{\text{old}}}(t^{(i)} \mid q, o)$. The objective is

$$\mathcal{J}(\phi) = \mathbb{E}\left[\frac{1}{G}\sum_{i=1}^{G} \min\big(\rho_i \hat{A}_i,\ \mathrm{clip}(\rho_i, 1-\varepsilon, 1+\varepsilon)\,\hat{A}_i\big) \ +\ \beta_{\text{ent}}\,\mathcal{H}\big[\pi_\phi(\cdot \mid q, o)\big]\right]. \tag{8}$$

We omit the reference-KL term and use an entropy bonus for cut exploration. Advantages computed within a rollout favor cuts with higher rescue rates than other cuts in that trace.

5.3 Gated Inference

At inference, ground-truth answers are unavailable, so we use a lightweight *gate* g_ψ to estimate whether each rollout is correct. Rollouts without a parseable `\boxed{}` answer receive a gate score of zero. For the rest, the same frozen Qwen3-0.6B encoder E processes the chat-templated question and rollout, and an MLP predicts correctness from the hidden state h_{box} at the final `\boxed{}` token. With κ denoting the calibration temperature, the gate is defined as

$$g_\psi(q, o) = \begin{cases} 0, & \text{if } o \text{ has no parseable answer,} \\ \sigma\big(\mathrm{MLP}_\psi(h_{\text{box}})/\kappa\big), & \text{otherwise,} \end{cases} \tag{9}$$

The gate is a 66K-parameter MLP trained once on 2,874 correctness-labeled rollouts and reused across all reasoners. Appendix D provides training and calibration details, evaluates gate ranking quality, compares learned and ground-truth gating, and examines threshold sensitivity.

Rollouts with $g_\psi(q, o) \geq \gamma$ are retained unchanged. Those below the fixed threshold $\gamma = 0.4$ are corrected once by sampling $t^\star \sim \pi_\phi(\cdot \mid q, o)$, injecting c via Eq. (6), and regenerating. Each regenerated rollout replaces its original, preserving the pool size.

6 Experiments

We ask four questions: (1) Can UPTURN correct rollouts to improve accuracy across models and benchmarks within a fixed generation budget? (2) How does it compare with self-correction and reflection-token injection? (3) Does it transfer across reasoners and domains and complement GRPO and self-correction? (4) How do learned cuts and turn-boundary injection affect performance?

6.1 Setup

Benchmarks, metrics, and reasoners. We train each position policy for 30 steps on DAPO-17K, a 17K-prompt DAPO-Math subset (Yu et al., 2025). Evaluation covers AIME24, AIME25, AIME26, AMC23, HMMT, and MATH500, all disjoint from training. **Avg** is the mean per-question accuracy of the final candidate pool and **Pass** the fraction of questions with at least one correct candidate. We evaluate four frozen reasoners in the main table: Qwen3-4B and Qwen3-8B with thinking enabled, DeepSeek-R1-0528-Qwen3-8B (DeepSeek-R1-Qwen3-8B), and QwQ-32B; two additional reasoners are reported in Appendix N. Unless stated otherwise, each reasoner uses a position policy trained on its own rollouts, while the 66K gate is reused unchanged across all models.

Table 1: Results on six math benchmarks. **Base@32** uses 32 rollouts; **UPTURN@16** uses 16 rollouts and corrects each gate-flagged one once. Δ denotes UPTURN's gain over Base; **bold** marks the higher score.

Method	AIME24		AIME25		AIME26		AMC23		HMMT		MATH500	
	Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass
Qwen3-4B												
Base@32	32.6	63.3	21.8	46.7	26.7	43.3	67.0	92.5	19.2	27.3	85.0	93.2
UPTURN@16	**50.8**	**83.3**	**39.0**	**56.7**	**44.2**	**70.0**	**84.2**	**95.0**	**28.0**	**39.4**	**91.0**	**95.8**
Δ	+18.2	+20.0	+17.2	+10.0	+17.5	+26.7	+17.2	+2.5	+8.8	+12.1	+6.0	+2.6
Qwen3-8B												
Base@32	35.8	56.7	20.9	43.3	24.7	50.0	64.8	92.5	19.0	27.3	83.1	93.0
UPTURN@16	**53.8**	**70.0**	**39.2**	**56.7**	**47.7**	**66.7**	**85.3**	**95.0**	**32.2**	**51.5**	**92.0**	**96.4**
Δ	+17.9	+13.3	+18.2	+13.3	+23.0	+16.7	+20.5	+2.5	+13.2	+24.2	+8.8	+3.4
DeepSeek-R1-Qwen3-8B												
Base@32	5.8	16.7	7.1	13.3	3.5	6.7	32.3	47.5	2.0	9.1	64.5	79.8
UPTURN@16	**43.8**	**70.0**	**31.2**	**53.3**	**35.2**	**56.7**	**79.5**	**97.5**	**24.6**	**48.5**	**82.8**	**95.6**
Δ	+37.9	+53.3	+24.2	+40.0	+31.7	+50.0	+47.2	+50.0	+22.6	+39.4	+18.3	+15.8
QwQ-32B												
Base@32	36.7	63.3	26.2	46.7	27.4	56.7	73.0	87.5	22.7	33.3	87.8	95.0
UPTURN@16	**54.4**	**70.0**	**33.8**	**56.7**	**49.4**	**63.3**	**86.2**	**92.5**	**27.1**	33.3	**92.2**	**96.2**
Δ	+17.7	+6.7	+7.5	+10.0	+22.0	+6.7	+13.2	+5.0	+4.4	0.0	+4.3	+1.2

Decoding budgets. Base and UPTURN share the frozen reasoner and sampling parameters. **Base@32** draws 32 independent full rollouts. **UPTURN@16** draws 16 rollouts and corrects each gate-flagged rollout once from $t^\star \sim \pi_\phi(\cdot|q, o)$, adding up to 16 generations. Thus, both methods allow up to 32 generations per question. Base therefore receives a larger maximum generated-token allowance: each correction reuses $o_{\le t^\star}$ and generates at most $L_{\max} - t^\star$ new tokens instead of a full rollout. Results at larger $L_{\max}$ are in Appendix I. Under gated inference (§5.3), corrections replace their originals, leaving 16 rollouts for evaluation. Sampling settings appear in Appendix C. We also measure end-to-end runtime, including gating and position-selection overhead (Appendix E).

6.2 Main Results

Table 1 compares **UPTURN@16** with **Base@32** across six benchmarks and four reasoners spanning model families and sizes from 4B to 32B. Despite Base receiving 32 full rollouts, UPTURN improves average accuracy on every benchmark and pass rate in all but one setting. This consistency supports the effectiveness of its correction mechanism. Gains are pronounced on AIME, whereas accuracy gains are smallest on MATH500 for every reasoner, where Base accuracy is already high.

The largest improvements occur on DeepSeek-R1-Qwen3-8B, reaching $+47.2$ points in average accuracy on AMC23 and $+53.3$ points in pass rate on AIME24. These gains highlight the effectiveness of redirecting existing rollouts under a constrained generation budget, where DeepSeek-R1-Qwen3-8B's long reasoning traces are frequently truncated. The benefits also persist on QwQ-32B: average accuracy improves on all six benchmarks, while pass rate improves on five and remains unchanged on HMMT. Results on two additional RL-finetuned reasoners show the same trend (Appendix N). We also assess robustness across three seeds for policy training and rollout sampling (Appendix M).

6.3 Comparison with Self-Correction Baselines

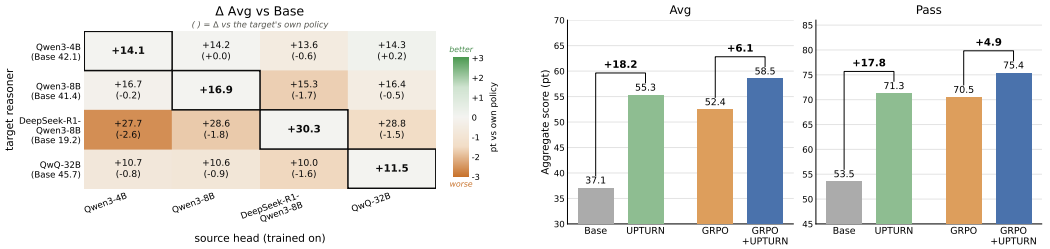
Table 2 compares UPTURN with self-correction methods Reflexion (Shinn et al., 2023) and SCoRe (Kumar et al., 2025), and reflection-token injection methods AlphaOne (Zhang et al., 2025) and CyclicReflex (Fan et al., 2026), averaged over Qwen3-4B, Qwen3-8B, and DeepSeek-R1-Qwen3-8B (full results in Appendices J and K). All methods start from the same 16 rollouts and correct each gate-flagged one once, following the budget protocol in Sec. 6.1. SCoRe conditions on the full first attempt, leaving almost no room for revision at $L_{\max}$, so we also compare methods at $2L_{\max}$.

UPTURN performs best at both budgets, with a larger advantage under the tighter token budget. At $L_{\max}$, it exceeds the strongest baseline, Reflexion, by $+16.2$ points in average accuracy and $+15.9$ in pass rate, while the two injection methods remain close to Base@32 (Table 1). Increasing the budget to $2L_{\max}$ improves the baselines evaluated at both budgets and narrows the gap, but UPTURN retains gains of $+4.9$ points in average accuracy and $+0.8$ in pass rate over the best baseline.

Table 2: **Comparison with other baselines.** Avg / Pass averaged over six benchmarks and all reasoners except QwQ-32B, at two budgets ($L_{\max}$ and $2L_{\max}$). Parentheses give UPTURN's gain over the best baseline at each budget; **bold** marks the best scores. †SCoRe requires $2L_{\max}$ as it conditions on the full first attempt.

	Self-correction		*Reflection-token injection*		
Budget	**Reflexion@16**	**SCoRe@16**	**AlphaOne@16**	**CyclicReflex@16**	**UPTURN@16**
$L_{\max}$	43.6 / 61.4	–†	32.6 / 45.2	35.7 / 49.7	**59.8 / 77.2** (+16.2 / +15.9)
$2L_{\max}$	55.3 / 77.5	52.3 / 79.9	55.0 / 71.8	57.2 / 73.5	**62.2 / 80.7** (+4.9 / +0.8)

6.4 Generalizability of UPTURN



(a) Position-policy transfer (b) UPTURN gains over Base and GRPO

Figure 4: **Position-policy transfer.** Averages over six benchmarks. **(a)** Avg gain over Base when the policy trained on the source reasoner is applied to the target; parentheses (and colour) give the difference from the target's own policy on the diagonal. **(b)** UPTURN gains over Base and GRPO, averaged over four reasoners; the GRPO setting reuses the corresponding Base-trained position policy without retraining.

Transfer across reasoners. Because the position policy uses a frozen encoder rather than reasoner-specific hidden states, we transfer it while keeping the target reasoner, gate, and correction pipeline fixed. Transfer incurs little performance loss (Figure 4a): averaged over all target–source pairs, transferred policies improve Avg over Base by $+17.2$ points, versus $+18.2$ for target-trained policies; no transfer falls more than 2.6 points below its target-trained counterpart. Relative to the gain over Base, this small gap suggests that the policy remains effective when reused across reasoners.

Transfer to another domain. UPTURN has so far been trained and tested only in mathematics. To assess domain generalisation, we evaluate on GPQA-Diamond (Rein et al., 2024), a benchmark of 198 graduate-level multiple-choice questions in physics, chemistry, and biology, under the protocol of Table 1 and without retraining, retuning, or changing the gate threshold on science data. Even so, UPTURN outperforms Base for every reasoner, as shown in Table 3.

Table 3: **Transfer to GPQA-Diamond**. Δ denotes UPTURN's gain over Base; **bold** marks the higher score. DS-R1 abbreviates DeepSeek-R1-Qwen3-8B.

Reasoner	**Base@32**		**UPTURN@16**		**Δ**	
	Avg	Pass	Avg	Pass	Avg	Pass
Qwen3-4B	32.5	56.6	**50.4**	**79.8**	+17.9	+23.2
Qwen3-8B	35.0	61.1	**54.3**	**83.8**	+19.3	+22.7
DS-R1	20.7	49.0	**35.9**	**78.8**	+15.2	+29.8
QwQ-32B	41.5	68.7	**54.2**	**80.8**	+12.7	+12.1

Complementarity with self-correction. We apply UPTURN once to gate-flagged solutions from each baseline's corrected pool, without retraining any component. Under a total response-token budget of $2L_{\max}$, UPTURN adds $+8.8$ Avg and $+3.0$ Pass points to Reflexion, and $+11.8$ Avg and $+1.8$ Pass points to SCoRe (Table 4). These gains show that UPTURN also complements existing self-correction methods. Full results are in Appendix L.

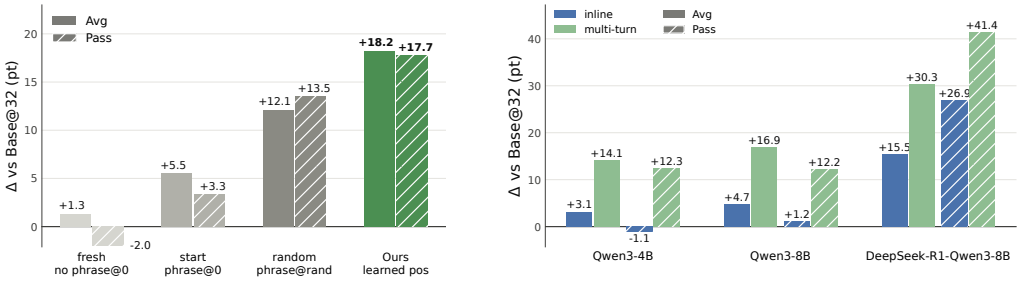
Table 4: **UPTURN after self-correction.** Avg / Pass averaged over six benchmarks and three reasoners (excluding QwQ-32B), at $2L_{\max}$. Δ gives the added gain; **bold** marks the better score in each pair.

Avg / Pass	**Reflexion@16**	**SCoRe@16**
Baseline	55.3 / 77.5	52.3 / 79.9
+ UPTURN	**64.1 / 80.5**	**64.1 / 81.8**
Δ	+8.8 / +3.0	+11.8 / +1.8

Complementarity with GRPO. Figure 4b tests whether UPTURN remains useful after reasoner post-training. Applying the corresponding Base-trained position policy to each GRPO-post-trained reasoner yields further gains of $+6.1$ Avg and $+4.9$ Pass points, averaged over four reasoners, without retraining the policy. These additional gains show that UPTURN complements GRPO.

Together, these results suggest that the learned position policy can support correction across reasoners, training stages, and domains.

6.5 Ablation



(a) Cut-position ablation (b) Inline vs. multi-turn injection

Figure 5: **Position and injection ablations.** Avg and Pass gains over Base@32, averaged over six benchmarks. **(a)** Fresh regeneration and injection at start, random, or learned positions, using the same gate and replacement rule; averages over four reasoners. **(b)** Inline and multi-turn injection with separately trained position policies.

Position: where to cut. All variants in Figure 5a use the same gate and replacement rule but differ in how regeneration is initiated. *Fresh* discards the flagged rollout and resamples it from the original prompt with the full budget $L_{\max}$; *start* injects the instruction at the very beginning of the rollout, so no prefix is retained; *random* injects it at a uniformly sampled paragraph boundary. Averaged over four reasoners and six benchmarks, the learned position performs best of the four variants.

Injection: learning the position for inline injection. One might suspect that inline injection fails in §4 because its position is not optimized. To test this possibility, we train a separate position policy for inline injection under exactly the training setup of §5.2, changing only the injection mode. As shown in Figure 5b, inline injection still yields smaller Avg and Pass gains than multi-turn injection for every reasoner. Thus, optimizing the injection position does not eliminate the gap, supporting the advantage of turn-boundary injection observed in §4. Appendix G further examines sensitivity to instruction wording, including an empty user turn with no injected text. We also examine whether additional correction rounds yield further gains (Appendix F).

Gate: selecting and flagging. We test the gate in two ways (Table 5). First, as a selector: we return the highest-gate candidate per question and report its accuracy (**GS**). Base@32 reaches 52.0 GS, close to its 53.5 Pass, so the gate reliably picks a correct rollout when one exists, and UPTURN@16 keeps a $+11.0$ point advantage under gate selection. Second, we flag with a ground-truth gate instead of the learned one. All metrics change by less than one point, so the learned gate is already close to optimal for flagging (details in Appendix D).

Table 5: **Gate ablations**, averaged over four reasoners and six benchmarks. **GS**: accuracy of the highest-gate candidate. +GT: flagging with a ground-truth gate.

Method	Avg	GS	Pass
Base@32	37.1	52.0	53.5
UPTURN@16	**55.3**	**63.0**	**71.3**
Δ	+18.2	+11.0	+17.7
UPTURN@16	55.3	63.0	**71.3**
UPTURN@16+GT	**55.9**	**63.3**	**71.3**
Δ	+0.6	+0.2	+0.1

7 Conclusion

We frame reasoning correction as redirecting an existing trajectory rather than retraining the reasoner. UPTURN builds on the observation that inline text is absorbed into the assistant's reasoning, whereas the same instruction delivered across a turn boundary can redirect it. A lightweight gate flags likely failures, a 130K-parameter head on a frozen encoder selects a cut that preserves a useful prefix, and the frozen reasoner regenerates from a fixed instruction injected as a new user turn. Across four reasoners and six math benchmarks, UPTURN consistently improves average accuracy and outperforms self-correction and reflection-token injection baselines under matched correction budgets, as well as heuristic cuts and inline injection. It transfers across reasoners and to GPQA-Diamond without retraining, and provides additional gains on top of GRPO and existing self-correction methods. When the required capability is already present, a small learned intervention can thus recover a better reasoning path without retraining the reasoner; extension to code and tool use remains open.

AI USE STATEMENT

In this work, we used generative AI tools to assist with editing the manuscript (formatting tables, revising text for clarity, and updating reported numbers from experiment logs) and as a coding assistant for analysis and plotting scripts. All AI-assisted text, numbers, and code were reviewed and verified by the authors: reported results were checked against the original experiment outputs, and analysis code was validated by reproducing previously verified results. Generative AI was not used to generate research ideas, related-work surveys, or experimental results. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024. doi:10.48550/arXiv.2407.21787.

Chongyu Fan, Yihua Zhang, Jinghan Jia, Alfred Hero, and Sijia Liu. Cyclicreflex: Improving reasoning models via cyclical reflection token scheduling. In *International Conference on Learning Representations*, volume 2026, pp. 109066–109088, 2026.

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, JD Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 54523–54549, 2025.

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.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Second Conference on Language Modeling*, 2025.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023. doi:10.52202/075280-2019.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321, November 2025. doi:10.18653/v1/2025.emnlp-main.1025.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. REFINER: Reasoning feedback on intermediate representations. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1100–1126, March 2024. doi:10.18653/v1/2024.eacl-long.67.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. In *Advances in Neural Information Processing Systems*, volume 37, pp. 55249–55285, 2024. doi:10.52202/079017-1754.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Liran Ringel, Elad Tolochinsky, and Yaniv Romano. Learning a continue-thinking token for enhanced test-time scaling. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 3324–3345, Mumbai, India, December 2025. doi:10.18653/v1/2025.ijcnlp-long.177.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. doi:10.48550/arxiv.1707.06347.

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.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. doi:10.52202/075280-0377.

Guiyao Tie, Zenghui Yuan, Zeli Zhao, Chaoran Hu, Tianhe Gu, Ruihang Zhang, Sizhe Zhang, Junran Wu, Xiaoyue Tu, Ming Jin, Qingsong Wen, Lixing Chen, Pan Zhou, and Lichao Sun. Can llms correct themselves? a benchmark of self-correction in llms. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, 2025. doi:10.52202/085713-5386.

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 *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 115452–115486, 2025. doi:10.52202/085713-3850.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. doi:10.52202/068431-1800.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi:10.48550/arXiv.2505.09388.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, juncai liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru 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, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 113222–113244, 2025. doi:10.52202/085713-3775.

Junyu Zhang, Runpei Dong, Han Wang, Xuying Ning, Haoran Geng, Peihao Li, Xialin He, Yutong Bai, Jitendra Malik, Saurabh Gupta, and Huan Zhang. AlphaOne: Reasoning models thinking slow and fast at test time. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 11329–11354, Suzhou, China, November 2025. doi:10.18653/v1/2025.emnlp-main.570.

A Illustrative Correction Example

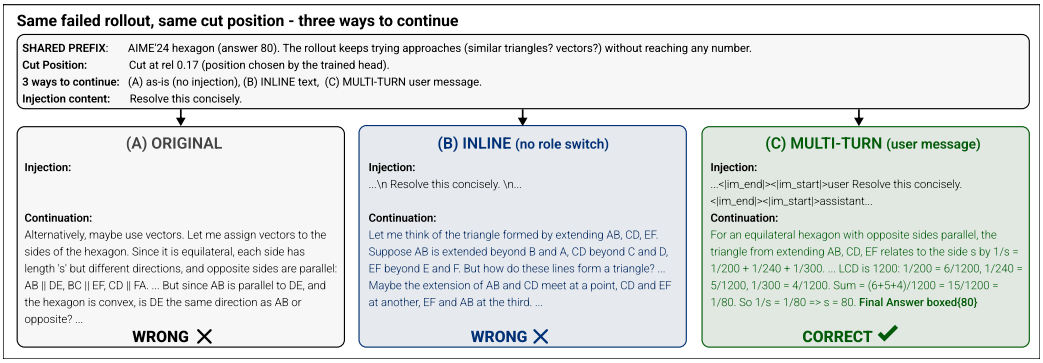


Figure 6: **One failed Qwen3-4B (Thinking on) rollout continued three ways from the same cut. (A)** Unmodified continuation remains trapped in re-verification. **(B)** Inline injection is absorbed as the model's own reasoning and resumes the loop. **(C)** The same instruction in a new user turn redirects reasoning to the correct answer. Full rollouts are provided in Appendix O.

B Limitations

UPTURN is a first step toward correcting reasoning by redirection rather than retraining, and several aspects remain open. Our study focuses on reasoning tasks whose outcomes can be checked automatically, and on a single, deliberately minimal form of intervention. Richer interventions, and settings where success is harder to verify, may call for different choices of what to inject and where. We leave these extensions to future work.

C Implementation Details

Position policy architecture. The position head f_ϕ operates on the 1024-dimensional last-layer hidden states of the frozen Qwen3-0.6B encoder. It applies LayerNorm to the input, followed by a linear layer $1024 \to 128$, a GELU activation, and a linear layer $128 \to 1$ that produces the logit of each candidate position, totaling 133,377 parameters (about 130K), the only trainable parameters of the position policy. The encoder receives exactly the token sequence the reasoner produced, the chat-templated prompt tokens concatenated with the rollout tokens, with no re-tokenization, as all our reasoners share the encoder's 151,643-token Qwen vocabulary. The delimiter `\n\n` is a single token in this tokenizer, and a failed rollout contains 11.3 candidate positions on average (median 5).

Training details. We train on DAPO-17K, a 17K-prompt subset of the DAPO-Math training pool. With dynamic sampling (at most 8 generation batches per step), each of the 30 training steps collects B=64 prompts containing at least one failed rollout, so training consumes 1,920 kept prompts in total. The head is updated once per step, sweeping all failed rollouts of the batch in mini-batches of 32 for one epoch, so the behavior policy $\pi_{\phi_{\text{old}}}$ is simply the policy that generated the step's data. Rollouts and regenerations use identical sampling settings. When a cut position falls inside an unclosed `<think>` block, we append `</think>` before injection so that the assistant turn remains well-formed, at both training and inference. Table 6 lists all hyperparameters.

Inference settings. The gate threshold is γ=0.4; Appendix D shows that results are robust to it. Each flagged rollout is corrected once and replaced by its regeneration, with no iterative rounds. The cut position is sampled from $\pi_\phi(\cdot \mid q, o)$.

Data separation. All evaluation benchmarks (AIME24/25/26, AMC23, HMMT, MATH500) are held-out test sets and do not overlap with the DAPO training pool. The gate's 1,197-prompt pool (Appendix D) is an independently sampled, topic-balanced subset of the same DAPO training pool.

Table 6: Hyperparameters

Hyperparameter	Value
Batch size B (prompts with $\geq$1 failure)	64
Rollouts per prompt N	8
Cut positions per rollout G	8
Continuations per position M	8
Generation budget $L_{\max}$	8192
Max generation batches (dynamic sampling)	8
Training steps	30
Mini-batch size / epochs per step	32 / 1
Optimizer	AdamW
Learning rate	1×10^{-3} (constant, no warmup)
Weight decay	0
Clipping range ε	0.2
Entropy coefficient β_{ent}	0.001
Sampling (rollout and regeneration)	temp. 0.6, top-p 0.95, top-k 20

Compute and software. All experiments run on NVIDIA B200 GPUs. The stack is a fork of verl, with vLLM 0.15.1 for generation, transformers 4.57.1, and PyTorch 2.9.1.

Instruction. The fixed instruction is $c =$ `"Resolve this concisely."`, used for all rollouts at both training and inference.

Multi-turn injection across model families. The turn boundary is realized with each reasoner's own chat-template tokens; Table 7 lists the sequence injected at the cut for each family.

Table 7: **Turn-boundary injection sequence per model family.** c denotes the fixed instruction.

Family (reasoners)	**Injected sequence at the cut**
Qwen ChatML (Qwen3-4B/8B, QwQ-32B)	`<|im_end|>\n<|im_start|>user\n` c `<|im_end|>\n<|im_start|>assistant\n`
DeepSeek (DeepSeek-R1-0528-Qwen3-8B)	`<|end_of_sentence|><|User|>` c `<|Assistant|><think>\n`

D GATE DETAILS

Training. The gate is trained once and reused unchanged across all experiments. Its training data comes from a single rollout pass, independent of the correction policy: we roll out 1,197 DAPO prompts, an independently sampled subset of the training pool in Appendix C, with Qwen3-4B-Instruct-2507 (16 samples per prompt), label each rollout with a valid final answer by correctness, and cache the hidden state h_{box} of the frozen encoder at the final `\boxed{}` token. Class-balancing yields 2,874 examples, on which the gate MLP, a 66K-parameter network separate from the 130K position head, is fit in seconds with the binary cross-entropy loss

$$\mathcal{L}_{\text{BCE}}(\psi) = -\frac{1}{n}\sum_{j=1}^{n}\Big[y_j \log \hat{p}_j + (1-y_j)\log\big(1-\hat{p}_j\big)\Big], \qquad \hat{p}_j = \sigma\big(\text{MLP}_\psi(h_{\text{box},j})\big), \tag{10}$$

where $y_j \in \{0,1\}$ is the correctness label. The temperature κ in Eq. (9) is calibrated post hoc on a held-out prompt split by minimizing BCE with the MLP weights frozen, giving $\kappa = 1.38$. The same gate is reused for every reasoner in our experiments, including thinking-mode models of different scales, since its features come from the frozen encoder reading the question and rollout text and are not tied to the generating model. The only non-trivial cost is the single rollout pass, so the gate is effectively free to fit, and in practice never needs refitting.

Ranking quality. Before any correction is generated, Table 8 measures how well the gate score ranks correct rollouts above incorrect ones, scored over the rollouts that produced an answer (a rollout that produces none is assigned gate 0 by convention and is never a ranking decision the gate has to make). Across the four reasoners the gate reaches a mean AUC of .962, with benchmark-wise AUC remaining at least .920.

Table 8: **Gate AUC by benchmark.** Mean over four reasoners.

Benchmark	AIME24	AIME25	AIME26	AMC23	HMMT	MATH500	**Avg**
AUC	.982	.983	.980	.984	.920	.922	**.962**

Learned versus ground-truth gating. The gate decides only *which* rollouts are corrected; it does not choose the cut, write the instruction, or regenerate. To see how much the reported numbers depend on that decision, we replace it with an oracle. In the ground-truth condition (+GT), a rollout is flagged exactly when its answer is wrong, instead of by the learned gate. Everything else is held fixed, so the oracle bounds what a perfect gate could buy.

Table 9 reports the comparison. Averaged over all four reasoners, the ground-truth gate changes Avg only from 55.3 to 55.9 and leaves Pass at 71.3. This small gap shows that the lightweight learned gate already identifies which rollouts require correction reliably, leaving little room for an oracle gate to improve the final result.

Table 9: **Learned (UPTURN@16) versus ground-truth (+GT) gating**, averaged over six benchmarks. **Bold** marks the higher score; ties are both bold.

Model	**Avg**			**Pass**		
	Base@32	UPTURN@16	UPTURN@16+GT	Base@32	UPTURN@16	UPTURN@16+GT
Qwen3-4B	42.1	56.2	**56.6**	61.1	73.4	**73.5**
Qwen3-8B	41.4	58.3	**58.7**	60.5	72.7	**72.8**
DeepSeek-R1-Qwen3-8B	19.2	49.5	**50.9**	28.8	**70.3**	**70.3**
QwQ-32B	45.7	57.2	**57.4**	63.8	**68.7**	**68.7**
Model Avg	37.1	55.3	**55.9**	53.5	**71.3**	**71.3**

Sensitivity to γ. Table 10 sweeps $\gamma \in [0, 1]$ over four reasoners and six benchmarks. At γ=0 no rollout is corrected and the pool is the uncorrected 16 rollouts; from γ=0.1 on, Avg varies by at most 0.4 points and Pass by at most 0.1, so the reported gains do not depend on tuning γ. Because gate scores concentrate near the extremes (mean AUC .962, Table 8), raising γ mainly sends already-confident rollouts to a redundant correction: the fraction corrected grows from 64% at γ=0.1 to 100% at γ=1.0 while accuracy stays essentially flat. We therefore set γ=0.4 as a cost–coverage compromise rather than by tuning for accuracy.

Table 10: **Sensitivity to the gate threshold γ.** Avg and Pass over four reasoners and six benchmarks; Corrected is the fraction of rollouts that receive a correction. **Bold** marks the default. At γ=0 nothing is corrected.

γ	0	0.1	0.2	**0.4**	0.6	0.8	1.0
Avg	37.3	55.6	55.5	**55.3**	55.2	55.2	55.2
Pass	51.8	71.2	71.3	**71.3**	71.3	71.3	71.3
Corrected (%)	0	64	69	**79**	91	99	100

Gate as a selector. Table 11 gives the per-benchmark results behind Table 5. Base@32 selects among its 32 rollouts and UPTURN@16 among the 16 candidates after correction, so both stay within the budget of Table 1; rollouts without an answer are never selected. UPTURN's GS gain over Base@32 is smaller than its Pass gain because corrected candidates are not always the highest-scored one, but UPTURN is higher in 21 of 24 cells, ties on AMC23 for both Qwen3 reasoners, and is lower only on HMMT for QwQ-32B.

Table 11: **Gate as a selector**, per benchmark. **GS** is the accuracy of the highest-gate candidate; **Avg** and **Pass** are as in Table 1. Δ denotes UPTURN's gain over Base; **bold** marks the higher score.

Method	**AIME24**			**AIME25**			**AIME26**		
	Avg	GS	Pass	Avg	GS	Pass	Avg	GS	Pass
Qwen3-4B									
Base@32	32.6	60.0	63.3	21.8	46.7	46.7	26.7	40.0	43.3
UPTURN@16	**50.8**	**66.7**	**83.3**	**39.0**	**53.3**	**56.7**	**44.2**	**53.3**	**70.0**
Δ	+18.2	+6.7	+20.0	+17.2	+6.7	+10.0	+17.5	+13.3	+26.7
Qwen3-8B									
Base@32	35.8	56.7	56.7	20.9	43.3	43.3	24.7	46.7	50.0
UPTURN@16	**53.8**	**60.0**	**70.0**	**39.2**	**46.7**	**56.7**	**47.7**	**63.3**	**66.7**
Δ	+17.9	+3.3	+13.3	+18.2	+3.3	+13.3	+23.0	+16.7	+16.7
DeepSeek-R1-Qwen3-8B									
Base@32	5.8	16.7	16.7	7.1	10.0	13.3	3.5	6.7	6.7
UPTURN@16	**43.8**	**40.0**	**70.0**	**31.2**	**36.7**	**53.3**	**35.2**	**50.0**	**56.7**
Δ	+37.9	+23.3	+53.3	+24.2	+26.7	+40.0	+31.7	+43.3	+50.0
QwQ-32B									
Base@32	36.7	60.0	63.3	26.2	43.3	46.7	27.4	56.7	56.7
UPTURN@16	**54.4**	**66.7**	**70.0**	**33.8**	**53.3**	**56.7**	**49.4**	**60.0**	**63.3**
Δ	+17.7	+6.7	+6.7	+7.5	+10.0	+10.0	+22.0	+3.3	+6.7

Method	**AMC23**			**HMMT**			**MATH500**		
	Avg	GS	Pass	Avg	GS	Pass	Avg	GS	Pass
Qwen3-4B									
Base@32	67.0	92.5	92.5	19.2	27.3	27.3	85.0	91.4	93.2
UPTURN@16	**84.2**	92.5	**95.0**	**28.0**	**33.3**	**39.4**	**91.0**	**92.4**	**95.8**
Δ	+17.2	0.0	+2.5	+8.8	+6.1	+12.1	+6.0	+1.0	+2.6
Qwen3-8B									
Base@32	64.8	92.5	92.5	19.0	27.3	27.3	83.1	91.4	93.0
UPTURN@16	**85.3**	92.5	**95.0**	**32.2**	**36.4**	**51.5**	**92.0**	**92.0**	**96.4**
Δ	+20.5	0.0	+2.5	+13.2	+9.1	+24.2	+8.8	+0.6	+3.4
DeepSeek-R1-Qwen3-8B									
Base@32	32.3	47.5	47.5	2.0	3.0	9.1	64.5	75.6	79.8
UPTURN@16	**79.5**	**90.0**	**97.5**	**24.6**	**27.3**	**48.5**	**82.8**	**89.8**	**95.6**
Δ	+47.2	+42.5	+50.0	+22.6	+24.2	+39.4	+18.3	+14.2	+15.8
QwQ-32B									
Base@32	73.0	87.5	87.5	22.7	**33.3**	33.3	87.8	93.0	95.0
UPTURN@16	**86.2**	**92.5**	**92.5**	**27.1**	30.3	33.3	**92.2**	**93.6**	**96.2**
Δ	+13.2	+5.0	+5.0	+4.4	-3.0	0.0	+4.3	+0.6	+1.2

E WALL-CLOCK COST

The fixed-budget protocol controls generation budgets, but does not guarantee equal wall-clock time. UPTURN requires two dependent generation stages, gate evaluation of all initial rollouts, and position selection and retained-prefix processing for flagged rollouts. These extra steps may offset the time saved by decoding fewer tokens. We therefore time both pipelines end to end on one idle NVIDIA B200 with Qwen3-4B on AIME24 at $L_{\max}$=8192. Each pipeline is executed twice; sampling is seeded, so generation and token counts are identical across the two repetitions. We report wall-clock time as the mean $\pm$ half the absolute difference between the two measurements. Base@32 generates all 32 rollouts per question in a single n=32 batch. Table 12 reports generated calls and decoded tokens per question and total wall-clock time for the benchmark, excluding the same 32 s engine start-up cost for every method.

Table 12: **Wall-clock cost on AIME24** (Qwen3-4B, one B200).

Method	Gens./q	Gen. tokens/q	Wall-clock (s)	vs. Base@32
Base@16	16.0	122,881	477 ± 2	$0.51\times$
Base@32	32.0	245,548	931 ± 0	$1.00\times$
UPTURN@16	**28.6**	**156,619**	$\mathbf{790 \pm 2}$	$\mathbf{0.85\times}$

UPTURN reduces decoded tokens by approximately 36% and wall-clock time by 15% relative to Base@32 as shown in Table 12. The wall-clock gain is smaller than the token gain because the overhead of additional gating, position-selection, and prefix-processing. However, these costs are smaller than the decoding time UPTURN saves by generating fewer tokens, so it remains faster overall.

F Does a Second Correction Round Help?

UPTURN corrects each flagged rollout only once. We examine whether further rounds improve performance on Qwen3-4B and Qwen3-8B across all six benchmarks, using up to three correction rounds under two rules. *Gated* accepts a regeneration only if it raises the gate score and stops correcting a rollout once its answer repeats; *naive* always replaces the current rollout and applies all three rounds. Here, *gated* refers to the acceptance and stopping rules, not the threshold used to select rollouts for correction. Both variants start from the same cached 16-rollout pool of Table 1, so round 0 is identical. We set γ=1.0 to remove threshold-based selection: all rollouts are initially eligible for correction, while subsequent rounds follow each variant's stopping rule. Because this protocol differs from the main evaluation, we focus on changes across rounds rather than absolute comparisons with Table 1.

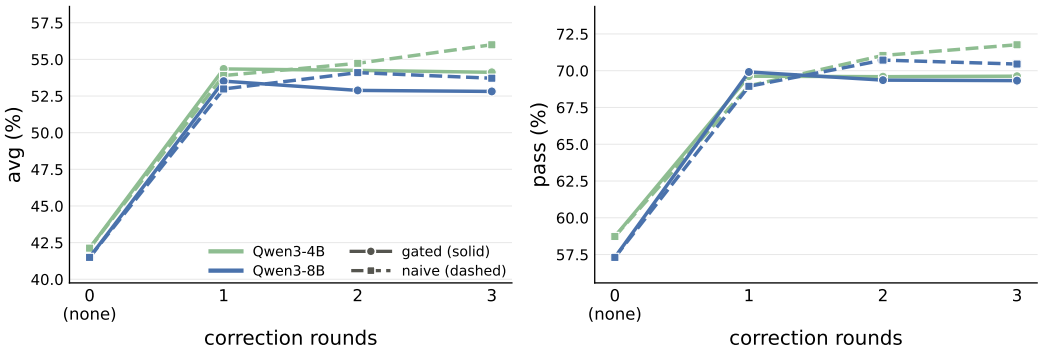


Figure 7: **Avg and Pass against the number of correction rounds**, averaged over six benchmarks. Colour marks the reasoner; solid lines are *gated*, dashed lines *naive*.

Figure 7 shows that the first correction accounts for most of the improvement under both rules. Under *gated* acceptance, additional rounds provide no further gains and slightly reduce Avg or Pass in some settings. At round 1, this rule achieves higher Avg and Pass than *naive* replacement on both reasoners. With further naive rounds, performance continues to improve on Qwen3-4B, while Qwen3-8B peaks at round 2 and declines slightly at round 3, remaining above round 1 on both metrics. Overall, additional rounds require further generation but offer smaller, rule- and reasoner-dependent benefits, motivating our single-round default.

G SENSITIVITY TO THE INJECTION INSTRUCTION

UPTURN injects a fixed instruction, "`Resolve this concisely.`", at the chosen cut. To separate the effect of the turn-boundary structure from that of the wording, we rerun the deployed pipeline on all six benchmarks varying only the injected text; the phase-1 pool, gate, position policy, budget, and seed are identical, so only the continuation after the cut differs. We compare four task-relevant instructions and three controls: the reflection token "`Wait.`", the meaningless "`Turtle.`", and *None*, an empty user turn in which only the chat-template turn-boundary tokens are inserted at the cut with no text at all.

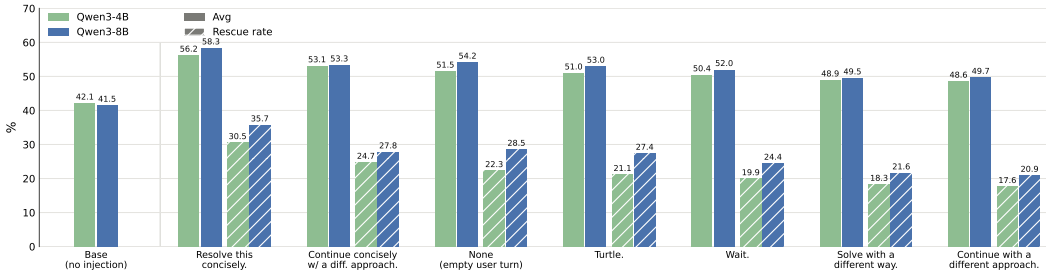


Figure 8: **Avg and rescue rate under varied injection instructions**, averaged over six benchmarks. Solid bars are Avg, hatched bars rescue rate; Base has no rescue rate. Instructions are ordered by mean Avg; *None* inserts the turn boundary with no text.

Figure 8 shows two effects. First, the injection structure matters most: every multi-turn variant improves Avg over Base on both reasoners, and even the empty user turn (*None*) reaches $51.5/54.2$ Avg on Qwen3-4B/8B against $42.1/41.5$ for Base. This is consistent with Figure 5b, where multi-turn injection outperforms inline injection by a wide margin on every reasoner. Second, once the injection is multi-turn, the wording matters: Avg across the seven variants spans 48.6 to 56.2 on Qwen3-4B and 49.5 to 58.3 on Qwen3-8B, and the default instruction improves Avg over *None* by 4.7 and 4.1 points, achieving the highest Avg and rescue rate on both reasoners. The turn boundary thus provides the structure that makes correction effective, and the instruction placed at it determines how much of that gain is realized.

H PARAGRAPH BOUNDARIES ARE FORKING POINTS

UPTURN restricts candidate cuts to `\n\n` boundaries. To check that this does not miss where a trace actually forks, we compute the next-token entropy at every position of the incorrect rollouts from the 16-rollout pool of Table 1 on AIME24/25, using each reasoner itself. Table 13 shows the same pattern for all four reasoners. Entropy is consistently high right after a paragraph boundary and near zero at a typical position, so every boundary is a forking point. Boundaries are only 2–3% of all positions (*boundary frac.*), yet they cover 18–35% of the most uncertain positions, the top-5% by entropy (*top-5% coverage*).

Table 13: **Next-token entropy at `\n\n` boundaries vs. elsewhere** on incorrect AIME24/25 rollouts; coverage is the share of top-5% entropy positions within ± 2 tokens of a boundary.

	Boundary entropy		**Other entropy**		**Top-5%**	**Boundary**
Reasoner	mean	median	mean	median	**coverage**	**frac.**
Qwen3-4B	**0.62**	**0.61**	0.29	0.008	18%	2.0%
Qwen3-8B	**0.60**	**0.59**	0.27	0.004	19%	2.0%
DeepSeek-R1-Qwen3-8B	**0.99**	**1.01**	0.31	0.018	35%	3.2%
QwQ-32B	**0.81**	**0.80**	0.41	0.095	22%	2.6%

A candidate set chosen by entropy would therefore contain the boundaries plus some extra positions inside paragraphs, while giving up two practical advantages: a `\n\n` cut always ends at a complete paragraph, so the turn boundary is grafted onto a well-formed prefix, whereas an entropy-chosen cut can fall in the middle of a sentence, and `\n\n` positions are found by string matching, whereas entropy-based candidates require an extra forward pass of the reasoner over every rollout at inference time. We therefore use `\n\n` boundaries as the candidate set of the position policy.

I SCALING THE GENERATION BUDGET

Figure 9 extends the comparison of Table 1 to per-rollout budgets from $L_{\max}$ to $3L_{\max}$ (8K to 24K tokens) on Qwen3-4B, Qwen3-8B, and DeepSeek-R1-Qwen3-8B across six benchmarks. Base@32 draws 32 rollouts at each budget, while UPTURN@16 starts from 16 rollouts and corrects each gate-flagged one once. Both methods allow up to 32 generations per question and use the same per-rollout length limit; for UPTURN, the retained prefix and regenerated suffix together remain within that limit. These constraints match the allowed budgets, not actual token usage. We apply the position policy trained at $L_{\max}$ unchanged at every budget, without retraining or adjusting the gate threshold, to evaluate its transfer to longer generation windows. Policies retrained for larger budgets are not evaluated here.

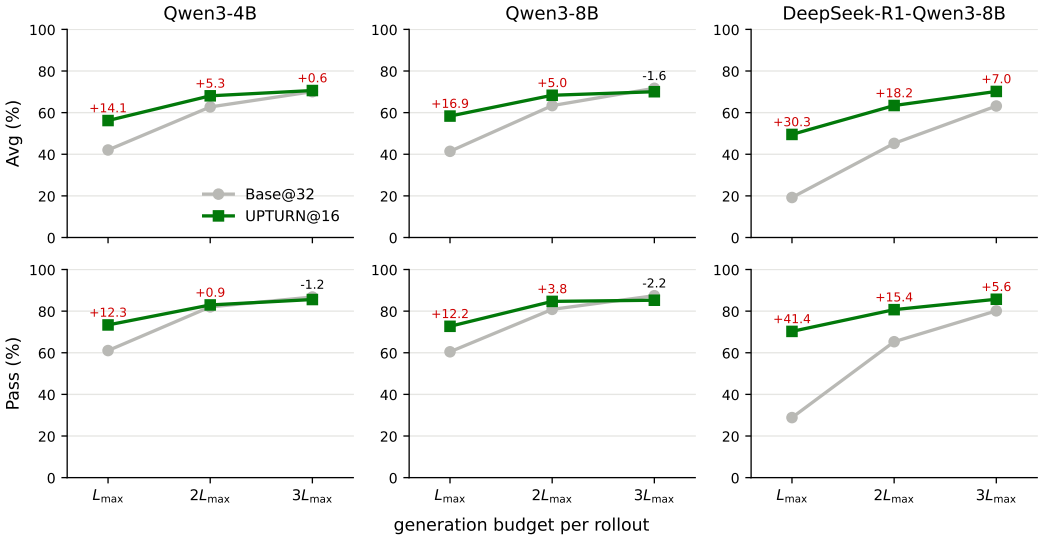


Figure 9: **Scaling the generation budget.** Avg (top) and Pass (bottom) averaged over six benchmarks as the per-rollout budget grows from $L_{\max}$ to $3L_{\max}$; labels give UPTURN's gain over Base at each budget.

UPTURN's advantage is largest at the smallest budget and narrows as the budget grows. At $L_{\max}$ it adds 14.1 to 30.3 Avg points across the three reasoners; at $2L_{\max}$, where Base@32 has already recovered more than 20 Avg points, it still adds $+5.3$, $+5.0$ and $+18.2$ points on Qwen3-4B, Qwen3-8B and DeepSeek-R1-Qwen3-8B. By $3L_{\max}$ the gap has closed on the two Qwen3 reasoners, where Base@32 matches or slightly exceeds UPTURN, while DeepSeek-R1-Qwen3-8B retains $+7.0$; Base@32 extends its lead at larger budgets. These results locate UPTURN's value under tight generation budgets, where it delivers its largest improvements without allowing longer reasoning traces.

J Full Comparison with Self-Correction Baselines

Table 14 breaks the self-correction rows of Table 2 down by reasoner and benchmark. Every row starts from the same 16 rollouts of Table 1 and corrects each gate-flagged one once; Budget is the number of new tokens a correction may generate. Reflexion (Shinn et al., 2023) reflects on the first attempt in a short note and then rewrites the solution from scratch, so it is run at both $L_{\max}$ and $2L_{\max}$. SCoRe (Kumar et al., 2025) conditions its second attempt on the full first attempt, so an $L_{\max}$ window leaves almost no room to write; we therefore report it at $2L_{\max}$ only. UPTURN continues from the retained prefix, and its continuation is granted the same $L_{\max}$ or $2L_{\max}$ window.

Table 14: **Full comparison with self-correction baselines**, per reasoner and benchmark, at $L_{\max}$ and $2L_{\max}$ correction budgets. Δ rows give UPTURN's gain over the best baseline at the same budget; **bold** marks the best entry per column and budget. SCoRe conditions on the full first attempt, so it is evaluated at $2L_{\max}$ only.

Method	Budget	AIME24		AIME25		AIME26		AMC23		HMMT		MATH500	
		Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass
Qwen3-4B													
Reflexion@16	$L_{\max}$	51.5	76.7	36.0	63.3	42.5	66.7	83.9	92.5	25.6	33.3	91.1	**96.6**
UPTURN@16	$L_{\max}$	**59.0**	**83.3**	**48.3**	**70.0**	**53.8**	**76.7**	**88.4**	**97.5**	**33.9**	**45.5**	**92.1**	96.4
Δ (vs. best)		+7.5	+6.7	+12.3	+6.7	+11.2	+10.0	+4.5	+5.0	+8.3	+12.1	+1.0	-0.2
Reflexion@16	$2L_{\max}$	**64.2**	**86.7**	49.0	76.7	55.6	80.0	90.2	**100.0**	34.3	**57.6**	**93.4**	**97.8**
SCoRe@16	$2L_{\max}$	47.3	**86.7**	44.8	**83.3**	47.3	**86.7**	64.2	**100.0**	29.7	48.5	59.6	97.0
UPTURN@16	$2L_{\max}$	62.9	**86.7**	**52.1**	80.0	**55.8**	80.0	**90.9**	97.5	**37.3**	**57.6**	92.7	96.8
Δ (vs. best)		-1.3	+0.0	+3.1	-3.3	+0.2	-6.7	+0.8	-2.5	+3.0	+0.0	-0.7	-1.0
Qwen3-8B													
Reflexion@16	$L_{\max}$	39.4	53.3	24.2	50.0	27.1	53.3	67.5	87.5	20.5	33.3	84.6	94.6
UPTURN@16	$L_{\max}$	**60.0**	**83.3**	**44.4**	**66.7**	**54.4**	**70.0**	**87.8**	**97.5**	**36.0**	**60.6**	**92.5**	**97.2**
Δ (vs. best)		+20.6	+30.0	+20.2	+16.7	+27.3	+16.7	+20.3	+10.0	+15.5	+27.3	+8.0	+2.6
Reflexion@16	$2L_{\max}$	45.8	76.7	33.3	73.3	34.2	73.3	70.3	**100.0**	26.7	51.5	85.7	97.0
SCoRe@16	$2L_{\max}$	63.1	**90.0**	**50.8**	**80.0**	54.0	**83.3**	72.8	**100.0**	36.7	60.6	67.9	97.4
UPTURN@16	$2L_{\max}$	**63.3**	83.3	49.6	**80.0**	**56.0**	76.7	**88.9**	97.5	**38.4**	**63.6**	**93.0**	**97.6**
Δ (vs. best)		+0.2	-6.7	-1.2	+0.0	+2.1	-6.7	+16.1	-2.5	+1.7	+3.0	+7.4	+0.2
DeepSeek-R1-Qwen3-8B													
Reflexion@16	$L_{\max}$	20.0	43.3	15.2	33.3	11.2	30.0	53.3	80.0	13.6	27.3	77.7	89.6
UPTURN@16	$L_{\max}$	**49.4**	**76.7**	**38.5**	**66.7**	**39.8**	**56.7**	**84.1**	**97.5**	**29.5**	**51.5**	**84.8**	**96.6**
Δ (vs. best)		+29.4	+33.3	+23.3	+33.3	+28.5	+26.7	+30.8	+17.5	+15.9	+24.2	+7.0	+7.0
Reflexion@16	$2L_{\max}$	45.4	**76.7**	36.0	60.0	36.9	56.7	84.1	**100.0**	23.5	36.4	**87.0**	95.4
SCoRe@16	$2L_{\max}$	45.0	73.3	32.3	53.3	37.1	56.7	80.3	97.5	26.7	48.5	80.9	95.6
UPTURN@16	$2L_{\max}$	**52.5**	**76.7**	**42.3**	**70.0**	**42.5**	**63.3**	**85.3**	97.5	**30.5**	**51.5**	85.2	**97.0**
Δ (vs. best)		+7.1	+0.0	+6.2	+10.0	+5.4	+6.7	+1.2	-2.5	+3.8	+3.0	-1.8	+1.4

Discussion. At $L_{\max}$, UPTURN beats Reflexion on every reasoner and benchmark in Avg, and on all but one Pass cell, by $+7.5$, $+18.6$ and $+22.5$ Avg on Qwen3-4B, Qwen3-8B and DeepSeek-R1-Qwen3-8B. Doubling the budget helps Reflexion the most, since a fresh rewrite is the method that most often runs out of room: at $2L_{\max}$ the two are within a point on Qwen3-4B, while UPTURN keeps a clear margin in Avg on Qwen3-8B and a smaller one on DeepSeek-R1-Qwen3-8B. SCoRe reaches a Pass comparable to Reflexion but a lower Avg: its rewrites are frequently truncated on the easier sets, which shows most clearly on MATH500 for Qwen3-4B and Qwen3-8B.

K Comparison with Reflection-Token Injection

Table 15 breaks the injection rows of Table 2 down by reasoner and benchmark. AlphaOne (Zhang et al., 2025) and CyclicReflex (Fan et al., 2026) steer the same trajectory by injecting reflection tokens inline, whereas UPTURN cuts it at a learned position and re-opens a user turn; none of the three trains the reasoner.

These are not correction methods but ways of steering a single generation, so we place them in UPTURN's correction slot: UPTURN's gate selects the rollouts of the 16-rollout pool of Table 1 that are likely wrong, and each method regenerates only those from the problem, aggregated with the same rule. The three therefore differ only in how a flagged rollout is redone. Budget is the number of new tokens a correction may generate, and every method is run at both $L_{\max}$ and $2L_{\max}$. At $L_{\max}$ the two injection methods stay close to Base@32: a regeneration that must also re-derive the discarded prefix rarely finishes within the window. With $2L_{\max}$ they improve by more than twenty points, yet UPTURN still leads at both budgets.

Table 15: **Full comparison with reflection-token injection**, per reasoner and benchmark, at $L_{\max}$ and $2L_{\max}$ correction budgets. Δ rows give UPTURN's gain over the best injection method at the same budget; **bold** marks the best entry per column and budget.

Method	Budget	AIME24		AIME25		AIME26		AMC23		HMMT		MATH500	
		Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass
Qwen3-4B													
AlphaOne@16	$L_{\max}$	33.8	53.3	21.9	36.7	27.3	46.7	67.0	87.5	16.9	24.2	84.9	92.4
CyclicReflex@16	$L_{\max}$	37.7	56.7	25.2	50.0	28.8	46.7	70.2	92.5	19.1	27.3	86.1	92.6
UPTURN@16	$L_{\max}$	**59.0**	**83.3**	**48.3**	**70.0**	**53.8**	**76.7**	**88.4**	**97.5**	**33.9**	**45.5**	**92.1**	**96.4**
Δ (vs. best)		+21.2	+26.7	+23.1	+20.0	+25.0	+30.0	+18.3	+5.0	+14.8	+18.2	+6.1	+3.8
AlphaOne@16	$2L_{\max}$	60.6	83.3	48.3	63.3	53.3	**80.0**	90.5	**100.0**	30.3	48.5	92.3	96.0
CyclicReflex@16	$2L_{\max}$	62.7	83.3	51.9	73.3	51.9	76.7	89.7	**100.0**	31.6	48.5	**92.7**	96.2
UPTURN@16	$2L_{\max}$	**62.9**	**86.7**	**52.1**	**80.0**	**55.8**	**80.0**	**90.9**	97.5	**37.3**	**57.6**	92.7	**96.8**
Δ (vs. best)		+0.2	+3.3	+0.2	+6.7	+2.5	+0.0	+0.5	-2.5	+5.7	+9.1	-0.1	+0.6
Qwen3-8B													
AlphaOne@16	$L_{\max}$	34.2	56.7	21.7	40.0	23.8	36.7	63.0	80.0	18.2	27.3	83.1	92.2
CyclicReflex@16	$L_{\max}$	36.2	60.0	21.9	36.7	27.1	43.3	66.2	87.5	19.3	27.3	84.1	92.2
UPTURN@16	$L_{\max}$	**60.0**	**83.3**	**44.4**	**66.7**	**54.4**	**70.0**	**87.8**	**97.5**	**36.0**	**60.6**	**92.5**	**97.2**
Δ (vs. best)		+23.8	+23.3	+22.5	+26.7	+27.3	+26.7	+21.6	+10.0	+16.7	+33.3	+8.4	+5.0
AlphaOne@16	$2L_{\max}$	62.5	**83.3**	48.5	66.7	52.9	**80.0**	87.7	**100.0**	33.7	48.5	92.7	96.6
CyclicReflex@16	$2L_{\max}$	**63.8**	80.0	48.8	70.0	55.0	76.7	87.5	97.5	34.3	57.6	93.0	97.0
UPTURN@16	$2L_{\max}$	63.3	**83.3**	**49.6**	**80.0**	**56.0**	76.7	**88.9**	97.5	**38.4**	**63.6**	**93.0**	**97.6**
Δ (vs. best)		-0.4	+0.0	+0.8	+10.0	+1.0	-3.3	+1.2	-2.5	+4.2	+6.1	+0.1	+0.6
DeepSeek-R1-Qwen3-8B													
AlphaOne@16	$L_{\max}$	4.0	10.0	3.8	10.0	2.7	6.7	26.9	40.0	0.2	3.0	53.8	70.6
CyclicReflex@16	$L_{\max}$	7.5	16.7	9.4	20.0	4.0	6.7	32.8	47.5	1.5	12.1	65.5	79.0
UPTURN@16	$L_{\max}$	**49.4**	**76.7**	**38.5**	**66.7**	**39.8**	**56.7**	**84.1**	**97.5**	**29.5**	**51.5**	**84.8**	**96.6**
Δ (vs. best)		+41.9	+60.0	+29.2	+46.7	+35.8	+50.0	+51.2	+50.0	+28.0	+39.4	+19.2	+17.6
AlphaOne@16	$2L_{\max}$	30.4	53.3	22.3	40.0	20.2	46.7	63.6	87.5	19.3	27.3	81.6	91.4
CyclicReflex@16	$2L_{\max}$	36.0	56.7	24.4	46.7	29.4	46.7	71.6	90.0	22.5	33.3	83.8	93.8
UPTURN@16	$2L_{\max}$	**52.5**	**76.7**	**42.3**	**70.0**	**42.5**	**63.3**	**85.3**	**97.5**	**30.5**	**51.5**	**85.2**	**97.0**
Δ (vs. best)		+16.5	+20.0	+17.9	+23.3	+13.1	+16.7	+13.8	+7.5	+8.0	+18.2	+1.4	+3.2

L Complementary with Other Baselines

UPTURN redirects a rollout rather than rewriting it, so it does not compete with a self-correction method for the same slot: it can run on whatever pool that method produces. Table 16 takes the 16 corrected solutions of Reflexion (Shinn et al., 2023) or SCoRe (Kumar et al., 2025) as the phase-1 pool, lets UPTURN's gate flag the weak ones among them, and redirects each flagged one once. Everything else, the gate, the position policy and the injected instruction, is unchanged from Table 1, and no component is retrained on the baseline's outputs. Reflexion is shown at both $L_{\max}$ and $2L_{\max}$, with UPTURN's continuation granted the same window as the pool it corrects. SCoRe conditions its second attempt on the full first attempt, so an $L_{\max}$ window leaves it almost no room to write and we report it at $2L_{\max}$ only.

Table 16: **UPTURN on top of a self-correction baseline**, per reasoner and benchmark. Each baseline produces its own 16 corrected solutions; UPTURN's gate then flags the weak ones among them and redirects each once, at the budget named on the baseline row. Reflexion is shown at both $L_{\max}$ and $2L_{\max}$; SCoRe conditions on the full first attempt, so only $2L_{\max}$ is meaningful for it. Δ rows give the gain of adding UPTURN; **bold** marks the better entry of the pair.

Method	AIME24		AIME25		AIME26		AMC23		HMMT		MATH500	
	Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass
Qwen3-4B												
Reflexion@16 ($L_{\max}$)	**51.5**	**76.7**	**36.0**	**63.3**	**42.5**	**66.7**	83.9	**92.5**	**25.6**	**33.3**	**91.1**	96.6
+ **UPTURN**	49.0	73.3	35.8	53.3	**42.5**	60.0	**84.4**	**92.5**	23.7	**33.3**	90.9	**97.0**
Δ	-2.5	-3.3	-0.2	-10.0	+0.0	-6.7	+0.5	+0.0	-1.9	+0.0	-0.3	+0.4
Reflexion@16 ($2L_{\max}$)	64.2	**86.7**	49.0	76.7	55.6	80.0	90.2	**100.0**	34.3	**57.6**	93.4	97.8
+ **UPTURN**	**69.0**	83.3	**56.7**	**80.0**	**61.9**	**83.3**	**92.3**	**100.0**	**36.2**	**57.6**	**93.5**	**98.0**
Δ	+4.8	-3.3	+7.7	+3.3	+6.2	+3.3	+2.2	+0.0	+1.9	+0.0	+0.2	+0.2
SCoRe@16 ($2L_{\max}$)	47.3	86.7	44.8	**83.3**	47.3	**86.7**	64.2	**100.0**	29.7	**48.5**	59.6	**97.0**
+ **UPTURN**	**66.9**	**90.0**	**56.7**	**83.3**	**59.8**	83.3	**92.2**	**100.0**	**37.3**	**48.5**	**92.5**	**97.0**
Δ	+19.6	+3.3	+11.9	+0.0	+12.5	-3.3	+28.0	+0.0	+7.6	+0.0	+32.9	+0.0
Qwen3-8B												
Reflexion@16 ($L_{\max}$)	39.4	53.3	24.2	**50.0**	27.1	53.3	67.5	87.5	20.5	33.3	84.6	94.6
+ **UPTURN**	**46.7**	**66.7**	**34.6**	**50.0**	**39.8**	**60.0**	**78.8**	**95.0**	**22.0**	**36.4**	**89.2**	**96.4**
Δ	+7.3	+13.3	+10.4	+0.0	+12.7	+6.7	+11.2	+7.5	+1.5	+3.0	+4.6	+1.8
Reflexion@16 ($2L_{\max}$)	45.8	76.7	33.3	73.3	34.2	73.3	70.3	**100.0**	26.7	51.5	85.7	97.0
+ **UPTURN**	**67.7**	**90.0**	**55.8**	**76.7**	**59.2**	**80.0**	**90.6**	**100.0**	**38.6**	**57.6**	**93.2**	**97.6**
Δ	+21.9	+13.3	+22.5	+3.3	+25.0	+6.7	+20.3	+0.0	+11.9	+6.1	+7.5	+0.6
SCoRe@16 ($2L_{\max}$)	63.1	**90.0**	50.8	**80.0**	54.0	**83.3**	72.8	**100.0**	36.7	60.6	67.9	97.4
+ **UPTURN**	**71.7**	**90.0**	**55.8**	76.7	**58.8**	**83.3**	**91.7**	**100.0**	**41.3**	**63.6**	**92.2**	**97.6**
Δ	+8.5	+0.0	+5.0	-3.3	+4.8	+0.0	+18.9	+0.0	+4.5	+3.0	+24.3	+0.2
DeepSeek-R1-Qwen3-8B												
Reflexion@16 ($L_{\max}$)	20.0	43.3	15.2	33.3	11.2	30.0	53.3	80.0	13.6	27.3	77.7	89.6
+ **UPTURN**	**39.0**	**66.7**	**29.8**	**46.7**	**31.9**	**50.0**	**69.5**	**95.0**	**18.8**	**39.4**	**78.3**	**94.2**
Δ	+19.0	+23.3	+14.6	+13.3	+20.6	+20.0	+16.2	+15.0	+5.1	+12.1	+0.6	+4.6
Reflexion@16 ($2L_{\max}$)	45.4	76.7	36.0	60.0	36.9	56.7	**84.1**	**100.0**	23.5	36.4	**87.0**	95.4
+ **UPTURN**	**54.0**	**80.0**	**45.8**	**63.3**	**45.6**	**63.3**	81.9	**100.0**	**29.9**	**42.4**	82.0	**96.6**
Δ	+8.5	+3.3	+9.8	+3.3	+8.8	+6.7	-2.2	+0.0	+6.4	+6.1	-5.0	+1.2
SCoRe@16 ($2L_{\max}$)	45.0	73.3	32.3	53.3	37.1	56.7	80.3	97.5	26.7	**48.5**	80.9	95.6
+ **UPTURN**	**52.9**	**83.3**	**40.4**	**63.3**	**44.2**	**66.7**	**84.4**	**100.0**	**29.7**	**48.5**	**84.6**	**96.4**
Δ	+7.9	+10.0	+8.1	+10.0	+7.1	+10.0	+4.1	+2.5	+3.0	+0.0	+3.7	+0.8

Adding UPTURN improves every baseline on every reasoner, averaged over benchmarks: $+6.6$ Avg and $+5.6$ Pass on Reflexion at $L_{\max}$, $+8.8$ and $+3.0$ on Reflexion at $2L_{\max}$, and $+11.8$ and $+1.8$ on SCoRe. The gain is largest where the pool it corrects is weakest. On Reflexion at $L_{\max}$, where a fresh rewrite most often runs out of room, it adds $+12.7$ Avg on DeepSeek-R1-Qwen3-8B against $+3.6$ on the two Qwen3 reasoners; doubling the budget reverses the ordering, to $+4.4$ on DeepSeek-R1-Qwen3-8B against $+11.0$ on Qwen3. Where the pool is already strong the gain narrows and can turn slightly negative, as on AMC23 and MATH500 for DeepSeek-R1-Qwen3-8B at $2L_{\max}$, on which Reflexion scores 84.1 and 87.0: a cut placed inside an already-correct rewrite can only lose. Redirecting a rollout therefore composes with rewriting it, and pays off most where the rewrite had the least room to finish.

M SEED ROBUSTNESS

Table 17 repeats Table 1 with three seeds, each used both to train the position policy and to draw the rollouts. The standard deviation of Avg stays within about two points, far below every gain of UPTURN over Base.

Table 17: **Seed robustness.** Table 1 repeated with three seeds for training the position policy and drawing rollouts; mean ± std over the three runs. Δ is UPTURN's gain; **bold** marks the higher mean.

Method	AIME24		AIME25		AIME26	
	Avg	Pass	Avg	Pass	Avg	Pass
Qwen3-4B						
Base@32	33.4±0.8	62.2±1.9	21.6±0.2	47.8±1.9	26.1±1.2	43.3±0.0
UPTURN@16	**52.1**±1.1	**81.1**±3.8	**40.0**±1.0	**58.9**±1.9	**45.8**±1.5	**70.0**±0.0
Δ	+18.7	+18.9	+18.4	+11.1	+19.7	+26.7
Qwen3-8B						
Base@32	35.0±1.0	61.1±5.1	21.3±0.6	43.3±3.3	25.3±0.6	46.7±3.3
UPTURN@16	**53.2**±0.6	**78.9**±7.7	**39.2**±0.9	**55.6**±1.9	**46.6**±1.3	**68.9**±1.9
Δ	+18.2	+17.8	+17.9	+12.2	+21.3	+22.2
DeepSeek-R1-Qwen3-8B						
Base@32	6.1±0.4	17.8±1.9	6.6±0.5	14.4±1.9	3.4±0.1	6.7±0.0
UPTURN@16	**41.5**±2.0	**64.4**±6.9	**31.5**±0.3	**53.3**±0.0	**33.4**±1.9	**57.8**±1.9
Δ	+35.3	+46.7	+25.0	+38.9	+30.0	+51.1

Method	AMC23		HMMT		MATH500	
	Avg	Pass	Avg	Pass	Avg	Pass
Qwen3-4B						
Base@32	68.3±1.2	92.5±0.0	19.5±0.5	26.3±1.7	84.9±0.1	93.5±0.3
UPTURN@16	**84.9**±1.3	**95.8**±1.4	**27.1**±1.3	**39.4**±3.0	**91.0**±0.2	**96.0**±0.3
Δ	+16.7	+3.3	+7.6	+13.1	+6.1	+2.5
Qwen3-8B						
Base@32	64.6±0.4	89.2±2.9	18.9±0.3	27.3±0.0	83.2±0.1	93.1±0.4
UPTURN@16	**84.5**±0.8	**95.8**±1.4	**31.6**±0.5	**47.5**±4.6	**91.8**±0.1	**96.5**±0.1
Δ	+20.0	+6.7	+12.7	+20.2	+8.7	+3.4
DeepSeek-R1-Qwen3-8B						
Base@32	32.2±0.3	49.2±2.9	2.0±0.2	13.1±3.5	64.5±0.0	80.6±0.8
UPTURN@16	**78.6**±0.9	**96.7**±1.4	**25.8**±1.1	**44.4**±4.6	**82.7**±0.5	**95.3**±0.2
Δ	+46.4	+47.5	+23.9	+31.3	+18.2	+14.7

N RESULTS ON OTHER MODELS

Table 18 reports results on two additional RL-finetuned thinking reasoners under the identical setup as Table 1: AceReason-Nemotron-1.1-7B and Skywork-OR1-7B. AceReason follows the Qwen ChatML template and Skywork the DeepSeek template, so the turn boundary is injected with the corresponding sequences of Table 7. On both, UPTURN improves average accuracy on every benchmark ($+1.8$ to $+15.2$ points) and pass rate on all but two benchmarks, where it ties Base.

Table 18: **Results on two additional reasoners**, with the same setup as Table 1. Δ denotes UPTURN's gain over Base; **bold** marks improvements.

Method	AIME24		AIME25		AIME26		AMC23		HMMT		MATH500	
	Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass	Avg	Pass
AceReason-Nemotron-1.1-7B												
Base@32	37.2	73.3	27.7	40.0	31.5	56.7	79.7	95.0	19.5	30.3	88.4	96.2
UPTURN@16	**47.5**	73.3	**32.5**	**46.7**	**44.6**	**63.3**	**89.7**	95.0	**27.1**	**45.5**	**92.1**	**97.0**
Δ	+10.3	0.0	+4.8	+6.7	+13.1	+6.7	+10.0	0.0	+7.6	+15.2	+3.7	+0.8
Skywork-OR1-7B												
Base@32	33.8	70.0	23.3	40.0	27.1	50.0	67.6	90.0	19.5	33.3	83.9	94.6
UPTURN@16	**49.0**	**73.3**	**34.2**	**56.7**	**40.0**	**66.7**	**81.1**	**95.0**	**27.3**	**42.4**	**85.7**	**97.6**
Δ	+15.2	+3.3	+10.8	+16.7	+12.9	+16.7	+13.5	+5.0	+7.8	+9.1	+1.8	+3.0

O FULL ROLLOUTS FOR FIGURE 6

PHASE-1 PREFIX (shared by A/B/C) p.1

PROBLEM: Let ABCDEF be a convex equilateral hexagon in which all pairs of opposite sides are parallel. The ...
PHASE-1 PREFIX -- shared by all three; (A)/(B)/(C) branch from the END of this prefix
(cut at rel 0.17, position chosen by the trained head)

<think> \n Okay, so I need to find the side length of this convex equilateral hexagon ABCDEF where all pairs of opposite sides are parallel. The problem also mentions that the triangle formed by extending the sides AB, CD, and EF has side lengths 200, 240, and 300. Hmm, let me try to visualize this. \n\n First, since it's an equilateral hexagon, all sides are equal. Let me denote the side length as 's'. Also, all pairs of opposite sides are parallel. That's an important condition. In regular hexagons, opposite sides are not only parallel but also equal and separated by some distance. However, this hexagon is equilateral but not necessarily regular, right? Because regular hexagons have all sides and angles equal, but here only the sides are equal, and opposite sides are parallel. So maybe it's something like a stretched or skewed regular hexagon? \n\n But the key here is that the triangle formed by extending AB, CD, and EF has sides 200, 240, 300. Let me try to understand how this triangle is formed. If I extend AB, CD, and EF, they should meet at some points to form a triangle. Wait, since opposite sides are parallel, maybe the extensions of AB, CD, EF meet at three different points, forming a triangle? \n\n Let me try to sketch this mentally. Let me consider the hexagon ABCDEF. Since opposite sides are parallel, AB is parallel to DE, BC is parallel to EF, and CD is parallel to FA. Wait, is that correct? Let me check. In a hexagon, opposite sides would be AB and DE, BC and EF, CD and FA? Wait, maybe I need to confirm the order. If it's labeled ABCDEF, then the opposite sides would be AB and DE, BC and EF, CD and FA? Hmm, maybe. Alternatively, sometimes in hexagons, opposite sides are AB and DE, BC and EF, CD and FA? Let me think. For example, in a regular hexagon, each side is opposite to the one three apart. So AB is opposite to DE, BC opposite to EF, CD opposite to FA. Yes, that seems right. \n\n So, if AB is parallel to DE, BC parallel to EF, CD parallel to FA. Now, the problem says that if we extend AB, CD, and EF, they form a triangle. Wait, but AB, CD, EF are three sides of the hexagon. If we extend them, maybe they meet at three different points? Let me think. If AB is extended, CD is extended, and EF is extended, maybe they meet at three different points to form a triangle. But since opposite sides are parallel, maybe the extensions of AB and DE are parallel, so they don't meet? But the problem says that extending AB, CD, EF form a triangle. So maybe not AB, CD, EF themselves, but their extensions? \n\n Alternatively, maybe the triangle is formed by the lines that contain AB, CD, and EF. Since AB is parallel to DE, CD is parallel to FA, EF is parallel to BC. But if we take the lines AB, CD, EF, they might not all meet at a single point. However, if we extend them, maybe they meet at three different points? Wait, but three lines in general position form a triangle. However, since AB is parallel to DE, and CD is parallel to FA, EF is parallel to BC. So maybe the lines AB, CD, EF are not parallel to each other? Let me think. \n\n But if AB is parallel to DE, CD is parallel to FA, EF is parallel to BC. So, AB is parallel to DE, which is not necessarily parallel to CD or EF. So, AB, CD, EF could be three lines that are not parallel to each other, hence forming a triangle when extended. So the triangle is formed by the three lines AB, CD, EF. Wait, but AB is a side of the hexagon. If we extend AB, CD, EF, they might meet at three different points, forming a triangle. The problem says that the triangle has sides 200, 240, 300. So, the lengths of the sides of this triangle are given. The question is to find the side length of the hexagon. \n\n So, perhaps there is a relationship between the side length of the hexagon and the sides of this triangle. Maybe similar triangles or something? \n\n Alternatively, maybe there is a formula or theorem related to equilateral hexagons with opposite sides parallel. I remember that in such hexagons, there's a relation to the triangle formed by the extensions of the sides. Maybe something like the side length of the hexagon is related to the sides of the triangle by some proportion? \n\n Alternatively, maybe using vectors or coordinate geometry. Let me try coordinate geometry. \n\n Let me try to assign coordinates to the hexagon. Let me assume that the triangle formed by the extensions of AB, CD, EF is a triangle with sides 200, 240, 300. Let me denote the triangle as triangle PQR, where P, Q, R are the intersections of the extended sides. But I need to figure out how the hexagon relates to this triangle. \n\n Alternatively, maybe the hexagon is inscribed in the triangle? Or maybe the triangle is formed by extending the sides of the hexagon. \n\n Alternatively, think about the fact that in such a hexagon, the opposite sides are parallel and the hexagon is equilateral. So, perhaps it's similar to a regular hexagon but scaled and sheared? \n\n Alternatively, think about the concept of a "parallelogram" in hexagons. Since opposite sides are parallel, maybe the hexagon can be thought of as a combination of parallelograms? \n\n Alternatively, maybe using affine transformations. Since affine transformations preserve ratios and parallelism, maybe we can transform the given triangle into a regular hexagon? But I don't know if that helps directly. \n\n Alternatively, think about the fact that in such a hexagon, the distances between the opposite sides are related to the sides of the triangle. \n\n Alternatively, think of the hexagon as a central triangle with three pairs of parallel sides. Wait, maybe not. \n\n Alternatively, think of the triangle formed by the extensions of AB, CD, EF as a triangle whose sides are related to the sides of the hexagon. \n\n Alternatively, maybe use the concept of similar triangles. If I can find similar triangles in the figure, then maybe the ratios can be used to find the side length. \n\n

(A) ORIGINAL (no injection)
-> Wrong

Alternatively, think about the hexagon as being formed by three pairs of parallel lines, each pair separated by some distance, and the sides of the triangle are related to those distances. \n\n Alternatively, maybe use vectors. Let me try to assign vectors to the sides of the hexagon. Since the hexagon is equilateral, each side has the same length 's', but different directions. However, opposite sides are parallel. So, for example, AB is parallel to DE, BC is parallel to EF, CD is parallel to FA. \n\n So, if I denote vectors AB, BC, CD, DE, EF, FA. Since AB is parallel to DE, so vector DE is a scalar multiple of AB. But since the hexagon is closed, the sum of vectors AB + BC + CD + DE + EF + FA = 0. But since AB is parallel to DE, and DE is the same direction as AB or opposite? Wait, since the hexagon is convex, the direction of DE should be same as AB? Wait, AB is a side from A to B, and DE is from D to E. Depending on the order of the hexagon, DE is next to EF, so maybe the direction is same as AB? Maybe not. Wait, maybe I need to think more carefully. \n\n Let me think of the hexagon ABCDEF in order. So, starting at A, then B, C, D, E, F, and back to A. So AB is from A to B, BC from B to C, etc. Then, DE is from D to E, EF from E to F, FA from F to A. \n\n Since opposite sides are parallel, AB is parallel to DE. So vector AB is parallel to vector DE. Similarly, BC is parallel to EF, and CD is parallel to FA. \n\n Since the hexagon is closed, the sum of all the vectors is zero: \n\n AB + BC + CD + DE + EF + FA = 0. \n\n But since DE is parallel to AB, and EF is parallel to BC, FA is parallel to CD. \n\n So, let me denote AB as vector **a**, BC as vector **b**, CD as vector **c**. Then DE is parallel to AB, so DE = k**a** for some scalar k. Similarly, EF is parallel to BC, so EF = m**b**, and FA is parallel to CD, so FA = n**c**. \n\n But since the hexagon is equilateral, the lengths of AB, BC, CD, DE, EF, FA are all equal to 's'. Therefore, the magnitude of **a**, **b**, **c** is 's', and the magnitude of k**a**, m**b**, n**c** is also 's'. Therefore, |k**a**| = |k| * |**a**| = |k| * s = s => |k| = 1. Similarly for m and n. Therefore, k, m, n are either 1 or -1. However, since the hexagon is convex, the directions of DE, EF, FA should be such that the hexagon remains convex. \n\n But let's think about the direction. If AB is vector **a**, then DE is parallel to AB. But since the hexagon is convex, DE should be in the same direction as AB? Or opposite? Let me think. If you go from D to E, which is the next side after DE. If AB is from A to B, then DE is from D to E. Since the hexagon is convex, the direction of DE should be same as AB? Maybe not necessarily. Let me try to think with coordinates. \n\n Alternatively, since the hexagon is closed, AB + BC + CD + DE + EF + FA = 0. Let me substitute DE = k**a**, EF = m**b**, FA = n**c**. Therefore: \n\n **a** + **b** + **c** + k**a** + m**b** + n**c** = 0. \n\n Grouping terms: \n\n (1 + k)**a** + (1 + m)**b** + (1 + n)**c** = 0. \n\n Since **a**, **b**, **c** are vectors in different directions (since the hexagon is convex and equilateral with opposite sides parallel), they are not colinear. Therefore, the coefficients must be zero: \n\n 1 + k = 0, \n\n 1 + m = 0, \n\n 1 + n = 0. \n\n Therefore, k = -1, m = -1, n = -1. \n\n Therefore, DE = -**a**, EF = -**b**, FA = -**c**. \n\n Therefore, the vectors: \n\n AB = **a**, \n\n BC = **b**, \n\n CD = **c**, \n\n DE = -**a**, \n\n EF = -**b**, \n\n FA = -**c**. \n\n Therefore, the hexagon is closed because **a** + **b** + **c** - **a** - **b** - **c** = 0. That checks out. \n\n So, the sides are AB = **a**, BC = **b**, CD = **c**, DE = -**a**, EF = -**b**, FA = -**c**. \n\n Now, since the hexagon is convex, the directions of **a**, **b**, **c** must be such that when you traverse the hexagon, the turns are all in the same direction (say, counterclockwise), and the vectors **a**, **b**, **c** are not colinear. \n\n Now, the triangle formed by extending AB, CD, EF. Let me think about how these lines intersect. \n\n AB is the line containing vector **a**. CD is the line containing vector **c**. EF is the line containing vector -**b**. Wait, EF is -**b**, so the direction of EF is opposite to **b**. So, the lines AB, CD, EF. \n\n But AB is in direction **a**, CD is in direction **c**, EF is in direction -**b**. So, these three lines are not necessarily parallel, so they can form a triangle. \n\n But how exactly? \n\n Let me think of the lines AB, CD, EF. Let me try to find their intersections. \n\n First, line AB: Let me parametrize it. Let me suppose that point A is at the origin, and vector AB is **a**. So, point A is (0,0), point B is **a**. \n\n Then, line AB is the line from A(0,0) in direction **a**. \n\n Similarly, line CD: Let me find coordinates for points C, D. \n\n Starting from point B, which is **a**, then vector BC is **b**, so point C is **a** + **b**. \n\n Then vector CD is **c**, so point D is **a** + **b** + **c**. \n\n Then line CD is from point D to E, which is D + (-**a**) = **a** + **b** + **c** - **a** = **b** + **c**. Wait, point E is D + DE = D + (-**a**) = **a** + **b** + **c** - **a** = **b** + **c**. Then, vector EF is -**b**, so point F is E + (-**b**) = **b** + **c** - **b** = **c**. Then, vector FA is -**c**, so point A is F + (-**c**) = **c** - **c** = 0, which matches. So, coordinates: \n\n A: 0 \n\n B: **a** \n\n C: **a** + **b** \n\n D: **a** + **b** + **c** \n\n E: **b** + **c** \n\n F: **c** \n\n So, the line AB is from A(0) to B(**a**). The line CD is from D(**a** + **b** + **c**) to E(**b** + **c**). The line EF is from E(**b** + **c**) to F(**c**). \n\n But the problem says that the triangle is formed by extending AB, CD, and EF. So, maybe the lines AB, CD, EF are extended beyond their segments to meet at three points forming a triangle. \n\n So, let me find the equations of these three lines. \n\n First, line AB: It's the line through points A(0) and B(**a**). So, parametric equations: t**a**, where t is a real number. \n\n Line CD: It's the line through points D(**a** + **b** + **c**) and E(**b** + **c**). So, direction vector is E - D = (**b** + **c**) - (**a** + **b** + **c**) = -**a**. Therefore, line CD can be parametrized as D + s(-**a**) = **a** + **b** + **c** - s**a**, where s is real. \n\n Line EF: It's the line through points E(**b** + **c**) and F(**c**). Direction vector is F - E = **c** - (**b** + **c**) = -**b**. So, parametrized as E + r(-**b**) = **b** + **c** - r**b**, where r is real. \n\n Now,

(B) INLINE (no role switch)
-> Wrong
\n Resolve this concisely. \n

Let me try to think of the hexagon as being formed by three pairs of parallel lines. Since AB is parallel to DE, BC parallel to EF, CD parallel to FA. Also, all sides are equal. \n\n If I consider the triangle formed by extending AB, CD, EF. Let me try to imagine that. Let me suppose that AB is extended beyond B and A, CD is extended beyond C and D, EF is extended beyond E and F. But how do these lines form a triangle? \n\n Alternatively, maybe the triangle is formed by the three lines that are not the sides of the hexagon but the extensions. For example, if you extend AB, CD, EF, they might meet at three different points, forming a triangle. So, for instance, the extension of AB and CD meet at a point, the extension of CD and EF meet at another point, and the extension of EF and AB meet at the third point. But since AB is parallel to DE, CD is parallel to FA, EF is parallel to BC, maybe these intersections form a triangle. \n\n Alternatively, perhaps the triangle is formed by the three lines that are the extensions of AB, CD, EF. So, for example, AB is extended in one direction, CD is extended in another direction, EF is extended in another direction, and these three lines form a triangle. \n\n But how can three lines form a triangle if they are not concurrent? Well, three lines in general position (no two parallel) form a triangle. However, in our case, AB, CD, EF are not necessarily parallel. Since AB is parallel to DE, CD parallel to FA, EF parallel to BC. But AB is not necessarily parallel to CD or EF. So, AB, CD, EF could be three lines that are not parallel, hence forming a triangle when extended. \n\n So, if we consider the triangle formed by AB, CD, EF, then each pair of lines intersect at a point, forming the triangle. The sides of this triangle are given as 200, 240, 300. So, the lengths of the sides of the triangle are 200, 240, 300. The problem is to find the side length of the hexagon. \n\n Now, how does the hexagon relate to this triangle? Maybe the hexagon is inscribed in the triangle? Or maybe the triangle is related to the hexagon through some scaling factor? \n\n Alternatively, think of the hexagon as being formed by cutting off triangles from the larger triangle. But since the hexagon is convex and equilateral, maybe there is a relation between the triangle and the hexagon. \n\n Alternatively, maybe using vectors. Let me try to assign coordinates. \n\n Let me consider the triangle formed by the lines AB, CD, EF. Let me denote this triangle as triangle T with sides 200, 240, 300. Let me assume that triangle T is such that AB, CD, EF are three of its sides? Wait, no. Wait, AB is a side of the hexagon, which is extended to form the triangle. So, the triangle is formed by extending AB, CD, EF, so the sides of the triangle are not the same as AB, CD, EF, but the lengths of the sides of the triangle are given as 200, 240, 300. \n\n Alternatively, maybe the triangle is similar to the triangle formed by the extensions. Wait, perhaps I need to think in terms of similar triangles or homothety. \n\n Alternatively, think about the concept of a "trapezoid" in the hexagon. Since opposite sides are parallel, the hexagon can be thought of as a combination of trapezoids or parallelograms. \n\n Alternatively, think of the hexagon as being formed by three pairs of parallel lines. Each pair of opposite sides is parallel, and all sides are equal. So, the hexagon can be considered as a centrally symmetric figure? Wait, not necessarily. In a regular hexagon, it is centrally symmetric, but in this case, since it's equilateral but not regular, maybe it's not centrally symmetric? However, if all opposite sides are parallel, then maybe it is centrally symmetric? Because if AB is parallel to DE, and BC is parallel to EF, and CD is parallel to FA, then maybe the hexagon is centrally symmetric with center at the intersection of the diagonals? \n\n Wait, maybe. If the hexagon is centrally symmetric, then for each vertex, there is an opposite vertex such that the center is the midpoint of the segment joining them. So, if that's the case, then maybe the hexagon can be thought of as a combination of parallelograms. \n\n Alternatively, maybe using the concept of vectors. Let me try to assign vectors to the sides of the hexagon. \n\n Let me denote the side length as $ s $. Let me assume that the hexagon is labeled in order A, B, C, D, E, F. Since opposite sides are parallel, the vector from A to B is the same as the vector from D to E, but in the opposite direction? Wait, no. Wait, if AB is parallel to DE, then vector AB is parallel to vector DE. Since AB is from A to B, and DE is from D to E. So, if AB is parallel to DE, then vector AB = k * vector DE for some scalar k. However, since the hexagon is equilateral, AB = DE in length. Therefore, vector AB is equal to vector DE. Wait, but direction? If AB is parallel to DE, but depending on the orientation, they could be in the same or opposite direction. \n\n But since the hexagon is convex, the direction should be consistent. Let me think. If AB is parallel to DE, then moving from A to B, then from D to E should be in the same direction? Or maybe opposite? If the hexagon is convex, then the direction of DE should be such that the overall shape is convex. \n\n Alternatively, maybe the vectors AB, BC, CD, DE, EF, FA are such that AB is parallel to DE, BC parallel to EF, CD parallel to FA. So, if AB is parallel to DE, then vector AB = vector DE. Similarly, vector BC = vector EF, vector CD = vector FA. But since the hexagon is closed, the sum of the vectors should be zero. \n\n So, AB + BC + CD + DE + EF + FA = 0. But since AB = DE, BC = EF, CD = FA, then substituting: \n\n AB + BC + CD + AB + BC + CD = 0 => 2(AB + BC + CD) = 0 => AB + BC + CD = 0. \n\n Therefore, AB + BC + CD = 0. So, this gives a relationship between the vectors. \n\n But since the hexagon is equilateral, the lengths of AB, BC, CD are all equal to 's', but their directions are different. However, their vector sum is zero. So, this is similar to a triangle, where three vectors add up to zero. Therefore, the three vectors AB, BC, CD form a triangle. Wait, but AB + BC + CD = 0 implies that AB + BC + CD = 0, so if you start at point A, go to B, then to C, then to D, and then the vector from D to A is equal to - (AB + BC + CD) = 0? Wait, no. Wait, AB + BC + CD = 0, so starting at A, moving AB to B, then BC to C, then CD to D, then from D, to get back to A, you need to move - (AB + BC

(C) MULTI-TURN (user message)
-> Correct (80)
<|im_end|><|im_start|>user Resolve this concisely.<|im_end|>...

<think> \n Okay, let me try to figure this out step by step. So, we have a convex equilateral hexagon ABCDEF with all pairs of opposite sides parallel. The triangle formed by extending AB, CD, and EF has sides 200, 240, and 300. We need to find the side length of the hexagon. \n\n First, since the hexagon is equilateral and opposite sides are parallel, it's a special type of hexagon. I remember that in such hexagons, the opposite sides are not only parallel but also the distances between them are related. Also, the triangle formed by extending the sides AB, CD, and EF is given. Let me think about how these sides relate. \n\n If we extend AB, CD, and EF, they should meet at three different points, forming a triangle. Let me denote the triangle as having sides of lengths 200, 240, and 300. The key is to relate the side length of the hexagon to these triangle sides. \n\n I recall that in such hexagons, the side length of the hexagon can be found by taking the reciprocal of the sum of the reciprocals of the sides of the triangle. Wait, is that correct? Or maybe it's related to the area or some other property? \n\n Alternatively, maybe there's a formula where the side length of the hexagon is the harmonic mean of the triangle sides? Let me check. \n\n Wait, let me think of a simpler case. Suppose the triangle formed by the extensions of the sides is similar to the triangle formed by the sides of the hexagon. But since the hexagon is equilateral, maybe there's a proportional relationship. \n\n Alternatively, think of the triangle as being formed by the extensions of the sides of the hexagon. Each side of the triangle is a line that is parallel to a side of the hexagon. Therefore, the triangle is similar to a triangle formed by the sides of the hexagon scaled by some factor. \n\n But since the hexagon is equilateral, all sides are equal. However, the triangle has sides of different lengths, so it's not similar. Therefore, maybe the sides of the triangle are related to the side length of the hexagon through some linear combination. \n\n Alternatively, consider that the triangle formed by the extensions of AB, CD, EF is called the "outer triangle" of the hexagon. In such cases, there's a formula that relates the side length of the hexagon to the sides of this triangle. \n\n I remember that in a convex equilateral hexagon with opposite sides parallel, the side length $ s $ of the hexagon is given by the formula: \n\n $$ \n \frac{1}{s} = \frac{1}{a} + \frac{1}{b} + \frac{1}{c} \n $$ \n\n But wait, that formula is for something else. Maybe not. Alternatively, maybe it's related to the reciprocal of the sum of the reciprocals of the triangle sides? \n\n Wait, if the triangle sides are 200, 240, 300, then maybe the side length of the hexagon is the harmonic mean? Let me compute that. \n\n The harmonic mean of 200, 240, 300 is: \n\n $$ \n \frac{3}{\frac{1}{200} + \frac{1}{240} + \frac{1}{300}} = \frac{3}{\frac{6}{1200} + \frac{5}{1200} + \frac{4}{1200}} = \frac{3}{\frac{15}{1200}} = \frac{3 \times 1200}{15} = \frac{3600}{15} = 240 \n $$ \n\n But that seems too straightforward. However, I need to verify if this formula is correct. Alternatively, maybe it's the reciprocal of the sum of the reciprocals? \n\n Wait, but the problem says that the triangle formed by extending AB, CD, EF has sides 200, 240, 300. Maybe the side length of the hexagon is related to the sides of this triangle through some proportional relationship. \n\n Alternatively, think of the triangle as being formed by three lines, each of which is parallel to a side of the hexagon. Since the hexagon is equilateral, the distances between these lines (the sides of the triangle) might be related to the side length of the hexagon. \n\n Alternatively, think of the triangle as being similar to the triangle formed by the sides of the hexagon, but scaled by some factor. However, since the hexagon is equilateral, the triangle formed by its sides would be equilateral, but in our case, the triangle is not equilateral, so that might not hold. \n\n Wait, here's an idea. In a convex equilateral hexagon with opposite sides parallel, the triangle formed by extending the sides AB, CD, EF is similar to the triangle formed by the sides of the hexagon. However, since the hexagon is equilateral, the triangle formed by the sides of the hexagon would be equilateral, but the given triangle is not. Therefore, this might not be the case. \n\n Alternatively, think of the hexagon as being formed by three pairs of parallel lines. Each pair of opposite sides is separated by some distance. The triangle formed by extending AB, CD, EF is related to these distances. \n\n Alternatively, think of the triangle as being the outer triangle of the hexagon, and the hexagon is inscribed within it. The side length of the hexagon can be related to the sides of the triangle through some geometric relations. \n\n Alternatively, think of the problem in terms of vectors. Let me try that approach. \n\n Let me denote the sides of the hexagon as vectors. Since the hexagon is equilateral, all sides have the same length $ s $, but different directions. However, opposite sides are parallel, so the vectors for AB and DE are equal, BC and EF are equal, CD and FA are equal. \n\n Let me denote the vectors: \n\n AB = vector $ \vec{a} $ \n\n BC = vector $ \vec{b} $ \n\n CD = vector $ \vec{c} $ \n\n DE = vector $ \vec{a} $ \n\n EF = vector $ \vec{b} $ \n\n FA = vector $ \vec{c} $ \n\n Since the hexagon is closed, the sum of the vectors should be zero: \n\n $ \vec{a} + \vec{b} + \vec{c} + \vec{a} + \vec{b} + \vec{c} = 0 $ \n\n Wait, that would be $ 2\vec{a} + 2\vec{b} + 2\vec{c} = 0 $, so $ \vec{a} + \vec{b} + \vec{c} = 0 $. Therefore, the three vectors $ \vec{a}, \vec{b}, \vec{c} $ form a triangle when added together. \n\n But how does this relate to the triangle formed by extending AB, CD, EF? \n\n The triangle formed by extending AB, CD, EF. Let me think about the lines AB, CD, EF. Since AB is parallel to DE, CD is parallel to FA, EF is parallel to BC. So, the lines AB, CD, EF are three lines that are not necessarily parallel to each other, and they form a triangle when extended. \n\n But the triangle formed by these lines has sides 200, 240, 300. So, maybe the sides of this triangle are related to the vectors $ \vec{a}, \vec{b},

the triangle is formed by the intersection of these three lines. So, we need to find the three intersection points of AB with CD, AB with EF, and CD with EF. \n\n Wait, but AB, CD, EF are three lines. Each pair of lines will intersect at a point. So, the triangle is formed by these three intersection points. \n\n So, let me find the three intersection points: \n\n First, intersection of AB and CD: Let me find t and s such that t**a** = **a** + **b** + **c** - s**a**. \n\n So, t**a** + s**a** = **a** + **b** + **c**. \n\n Therefore, (t + s)**a** = **a** + **b** + **c**. \n\n But since **a**, **b**, **c** are not colinear (as the hexagon is convex), the only way this equation holds is if the coefficients of **a**, **b**, **c** are equal on both sides. However, the left side has only **a** term, while the right side has **a**, **b**, **c**. Therefore, unless **b** and **c** are zero, which they are not, this equation can't hold. Therefore, maybe I need to think again. \n\n Wait, maybe I need to parametrize the lines differently. Let me think again. \n\n Line AB is parametrized as t**a**, where t in ?. \n\n Line CD is parametrized as **a** + **b** + **c** - s**a**, where s in ?. \n\n So, setting them equal: \n\n t**a** = **a** + **b** + **c** - s**a** \n\n => (t + s)**a** = **a** + **b** + **c** \n\n But since **a**, **b**, **c** are linearly independent? Wait, no, they are not necessarily linearly independent. Wait, in a hexagon, the vectors **a**, **b**, **c** are not necessarily independent. However, since the hexagon is closed, we had earlier that **a** + **b** + **c** - **a** - **b** - **c** = 0, which is trivial. But in reality, **a**, **b**, **c** are vectors in the plane, so they are linearly dependent if they are in 2D. Wait, but in 2D, three vectors are linearly dependent. However, in our case, since the hexagon is convex and equilateral with opposite sides parallel, maybe **a**, **b**, **c** are not colinear, but they are in the plane, so they must satisfy some relation. \n\n But how can we solve for t and s? Let me think. Let me write the equation: \n\n (t + s)**a** = **a** + **b** + **c** \n\n So, this implies that **a** + **b** + **c** must be a scalar multiple of **a**, but since **b** and **c** are not parallel to **a** (otherwise, the hexagon would be degenerate), this equation can't hold unless **b** and **c** are zero, which they are not. Therefore, this suggests that lines AB and CD do not intersect? But that can't be, since they are two lines in the plane, they should intersect unless they are parallel. \n\n But AB is in direction **a**, and CD is in direction -**a**, so they are parallel? Wait, line AB is direction **a**, and line CD is direction -**a**. Therefore, they are parallel. Therefore, they do not intersect unless they are the same line. But AB is from A(0) to B(**a**), while CD is from D(**a** + **b** + **c**) to E(**b** + **c**). Since AB is in direction **a**, and CD is in direction -**a**, they are parallel lines. Therefore, they don't intersect unless they are the same line, which they are not. Therefore, the lines AB and CD are parallel, so they don't meet. Therefore, my previous assumption is wrong. \n\n But the problem says that the triangle is formed by extending AB, CD, EF. So, perhaps I need to extend AB and CD beyond their segments? Wait, but they are parallel, so they don't meet. Therefore, maybe the triangle is formed by extending AB, CD, and EF in some other way? Maybe not AB, CD, EF, but their extensions? \n\n Wait, maybe the triangle is formed by extending AB, CD, and EF beyond certain points. Wait, but if AB and CD are parallel, then they can't form a triangle with EF. So, maybe I misunderstood which sides are being extended? \n\n Wait, the problem says: "the triangle whose sides are extensions of segments AB, CD, and EF". So, the triangle has sides that are extensions of AB, CD, and EF. So, maybe the triangle is formed by the three lines that contain AB, CD, EF. But since AB and CD are parallel, they can't form a triangle. Therefore, there must be something wrong with my reasoning. \n\n Wait, maybe the triangle is formed by extending AB, CD, and EF not all three, but in a different way? Or maybe the triangle is formed by the intersections of AB with CD extended, AB with EF extended, and CD with EF extended? But if AB and CD are parallel, then AB and CD can't intersect. Therefore, there must be a mistake in my earlier analysis. \n\n Wait, maybe my assumption that DE = -**a** is incorrect? Let me check again. \n\n Earlier, I had that the sum of vectors AB + BC + CD + DE + EF + FA = 0. Then, I assumed that DE is parallel to AB, so DE = k**a**, EF = m**b**, FA = n**c**, and since the lengths are equal, k, m, n are ?1. Then, substituting into the equation gives (1 + k)**a** + (1 + m)**b** + (1 + n)**c** = 0. Therefore, if the hexagon is closed, the coefficients must be zero. Therefore, k = m = n = -1. Therefore, DE = -**a**, EF = -**b**, FA = -**c**. \n\n But if that is the case, then AB is in direction **a**, DE is in direction -**a**, so they are parallel. Similarly, BC is **b**, EF is -**b**, so they are parallel, and CD is **c**, FA is -**c**, so they are parallel. So that's correct. \n\n Therefore, lines AB and DE are parallel, lines BC and EF are parallel, lines CD and FA are parallel. Therefore, lines AB and CD are not necessarily parallel. Wait, AB is direction **a**, CD is direction **c**. So, unless **a** and **c** are parallel, which they are not in a general hexagon. Therefore, lines AB and CD are not parallel. Similarly, lines AB and EF: AB is direction **a**, EF is direction -**b**. So, unless **a** and -**b** are parallel, which they are not, so they are not parallel. Similarly, CD and EF: direction **c** and -**b**, not parallel. Therefore, the three lines AB, CD, EF are not parallel, so they should form a triangle when extended. \n\n But earlier, when I tried to find the intersection of AB and CD, I found that they are parallel? Wait, no, AB is direction **a**, CD is direction **c**, so they are not parallel. Wait, but in my previous calculation, I thought that line CD is direction -**a**, but that was a mistake. \n\n Wait, let me re-examine the parametrization of line CD. Earlier, I thought that line CD is from D to E, which is direction vector E - D = (**b** + **c**) - (**a** + **b** + **c**) = -**a**. Wait, that is correct. Therefore, line CD is in direction -**a**. But line AB is in direction **a**, so they are parallel. Therefore, lines AB and CD are parallel. Therefore, they cannot form a triangle with EF. Therefore, something is wrong here. The problem states that the triangle is formed by extending AB, CD, EF. So, how? \n\n Wait, maybe the triangle is not formed by the lines AB, CD, EF, but by the lines that + CD) = 0. Wait, that would mean that D is equal to A? That can't be. Wait, I think I made a mistake here. \n\n Wait, the sum of all six vectors is zero: AB + BC + CD + DE + EF + FA = 0. Since AB = DE, BC = EF, CD = FA, then substituting: \n\n AB + BC + CD + AB + BC + CD = 0 => 2(AB + BC + CD) = 0 => AB + BC + CD = 0. \n\n So, AB + BC + CD = 0. Therefore, if you start at point A, go along AB to B, then BC to C, then CD to D, then the vector from D to A is - (AB + BC + CD) = 0. Therefore, D is equal to A? That can't be. Wait, that seems contradictory. Wait, no, the vector from D to A is - (AB + BC + CD). But AB + BC + CD = 0, so the vector from D to A is zero, meaning that D = A? That can't be, since D is a different vertex. Therefore, there must be a mistake here. \n\n Wait, perhaps I made a mistake in the direction of the vectors. Let me check. The vector AB is from A to B, vector BC is from B to C, vector CD is from C to D, vector DE is from D to E, vector EF is from E to F, vector FA is from F to A. \n\n Therefore, the sum AB + BC + CD + DE + EF + FA = 0. \n\n Since AB is parallel to DE, and AB = DE in length, but direction? If AB is parallel to DE, then DE is in the same direction as AB? Or opposite? \n\n If the hexagon is convex, then the direction of DE should be such that the hexagon remains convex. Let me think. If AB is going in some direction, then DE, being opposite to AB, should be in the opposite direction? Wait, but if AB is parallel to DE, then DE is in the same direction as AB? Or opposite? If the hexagon is convex, then moving from D to E should be in the same direction as AB? Maybe not. Maybe it's in the opposite direction? \n\n Wait, let me think of a regular hexagon. In a regular hexagon, each side is parallel to the one three sides away. For example, AB is parallel to DE, but in the regular hexagon, AB is going in one direction, and DE is going in the same direction. So, vector AB is equal to vector DE. Therefore, in that case, AB + BC + CD + DE + EF + FA = 0. If AB = DE, BC = EF, CD = FA, then AB + BC + CD + AB + BC + CD = 0 => 2(AB + BC + CD) = 0 => AB + BC + CD = 0. However, in a regular hexagon, AB + BC + CD is not zero. Wait, in a regular hexagon, AB + BC + CD is a vector that points from A to D, but since it's regular, the sum of AB + BC + CD would be equal to AD, which is not zero. Therefore, this seems contradictory. Therefore, my previous assumption that AB = DE is wrong? \n\n Wait, in a regular hexagon, AB is parallel to DE, but they are not equal in direction? Wait, in a regular hexagon, AB is from A to B, and DE is from D to E. In a regular hexagon, moving from D to E is the same direction as AB. So, vector AB is equal to vector DE. However, in that case, AB + BC + CD + DE + EF + FA = 0. But AB + BC + CD + DE + EF + FA = AB + BC + CD + AB + BC + CD = 2(AB + BC + CD) = 0. Therefore, AB + BC + CD = 0. But in a regular hexagon, AB + BC + CD is equal to AD. But in a regular hexagon, AD is twice the length of AB, but in direction? Wait, if AB is a vector, then AD is a vector that is twice AB? No, in a regular hexagon, the distance from A to D is twice the length of AB? Wait, no. In a regular hexagon, the distance between opposite vertices is twice the side length. But AD is a diagonal, not a side. Wait, in a regular hexagon, AB is a side, BC is a side, CD is a side, DE is a side, EF is a side, FA is a side. The vector AB + BC + CD is equal to vector AD. In a regular hexagon, vector AD is twice the vector AB? No, because in a regular hexagon, the angle between AB and BC is 120 degrees. So, adding vectors AB + BC + CD is not straightforward. Wait, maybe I need to calculate it. \n\n Let me assign coordinates to a regular hexagon for clarity. Let me take a regular hexagon with side length 1, centered at the origin. Let me assign coordinates: \n\n Let me consider point A at (1, 0). Then, moving counterclockwise, point B is at (cos(60?), sin(60?)) = (0.5, sqrt3/2), point C at (-0.5, sqrt3/2), point D at (-1, 0), point E at (-0.5, -sqrt3/2), point F at (0.5, -sqrt3/2), and back to A. \n\n So vector AB is from A(1,0) to B(0.5, sqrt3/2): (-0.5, sqrt3/2). \n\n Vector BC is from B(0.5, sqrt3/2) to C(-0.5, sqrt3/2): (-1, 0). \n\n Vector CD is from C(-0.5, sqrt3/2) to D(-1, 0): (-0.5, -sqrt3/2). \n\n Adding AB + BC + CD: (-0.5, sqrt3/2) + (-1, 0) + (-0.5, -sqrt3/2) = (-0.5 -1 -0.5, sqrt3/2 + 0 - sqrt3/2) = (-2, 0). Which is the vector from A to D, which is (-2, 0). Therefore, AB + BC + CD is equal to vector AD, which is not zero. However, according to the previous conclusion, AB + BC + CD = 0. But in reality, in the regular hexagon, AB + BC + CD = AD ? 0. Therefore, my previous conclusion that AB + BC + CD = 0 is wrong. Therefore, there must be a mistake in my reasoning. \n\n Wait, where is the mistake? Earlier, I said that AB = DE, BC = EF, CD = FA, so substituting into the sum AB + BC + CD + DE + EF + FA = 0 gives 2(AB + BC + CD) = 0. But in reality, in the regular hexagon, AB is equal to DE, BC equal to EF, CD equal to FA, but AB + BC + CD is not zero. Therefore, my mistake is in assuming that DE = AB, EF = BC, FA = CD. But in reality, DE is equal in length to AB, but in direction? In the regular hexagon, vector DE is from D(-1, 0) to E(-0.5, -sqrt3/2): (0.5, -sqrt3/2). Which is the negative of vector AB, which was (-0.5, sqrt3/2). Therefore, vector DE = - vector AB. Similarly, vector EF is from E(-0.5, -sqrt3/2) to F(0.5, -sqrt3/2): (1, 0), which is - vector BC, since vector BC was (-1, 0). Similarly, vector FA is from F(0.5, -sqrt3/2) to A(1, 0): (0.5, sqrt3/2), which is - vector CD. Therefore, in the regular hexagon, DE = - AB, EF = - BC, FA = - CD. Therefore, substituting into the sum AB + BC + CD + DE + EF + FA = AB + BC + CD - AB - BC - CD = 0. Ah! Therefore, the correct relationship is that DE = - AB, EF = - BC, FA = - CD. Therefore, the previous mistake was assuming that DE = AB, but actually DE = - AB. Therefore, in general, for the hexagon, since opposite sides are parallel, DE is parallel to AB but in the opposite direction. Therefore, vector DE = - vector AB, similarly for others. Therefore, the sum AB + BC + CD + DE + EF + FA = AB + BC + CD - AB - BC - CD = 0, which checks out. \n\n Therefore, in general, for this hexagon, the vectors AB, BC, CD are such that DE = - AB, EF = - BC, FA = - CD. Therefore, the hexagon is closed. \n\n So, in this case, the hexagon can be thought of as having three vectors AB, BC, CD, and then the other three sides are their negatives. \n\n Now, the problem says that the triangle formed by extending AB, CD, EF has sides 200, 240, 300. Let me think about how this triangle is related to the hexagon. \n\n Since AB, CD, EF are sides of the hexagon, and they are extended to form a triangle. Let me try to think of the triangle as follows: suppose we extend AB beyond B, CD beyond D, and EF beyond F, and these three lines meet at three different points, forming \vec{c} $? \n\n Alternatively, since $ \vec{a} + \vec{b} + \vec{c} = 0 $, the triangle formed by these vectors is a triangle with sides $ |\vec{a}|, |\vec{b}|, |\vec{c}| $. But in our case, the hexagon is equilateral, so $ |\vec{a}| = |\vec{b}| = |\vec{c}| = s $. Therefore, the triangle formed by vectors $ \vec{a}, \vec{b}, \vec{c} $ is an equilateral triangle with side length $ s $. But the triangle formed by extending AB, CD, EF is not equilateral. Therefore, this seems conflicting. \n\n Wait, but maybe the triangle formed by the extensions is not the triangle formed by the vectors $ \vec{a}, \vec{b}, \vec{c} $, but another triangle. Maybe the triangle formed by the lines AB, CD, EF. \n\n Let me think about the lines AB, CD, EF. Since AB is parallel to DE, CD is parallel to FA, EF is parallel to BC. If I extend AB, CD, EF, they will meet at three different points, forming a triangle. Let me denote the intersection of AB and CD as point P, CD and EF as Q, and EF and AB as R. Wait, but AB and CD might not intersect unless extended. Since the hexagon is convex, the sides AB and CD are not adjacent, so their extensions would meet at some point. \n\n But since the hexagon is convex and opposite sides are parallel, the triangle formed by extending AB, CD, EF is actually the triangle formed by the three lines AB, CD, EF. Let me think of this triangle as triangle PQR, where P is the intersection of AB and CD, Q is the intersection of CD and EF, and R is the intersection of EF and AB. \n\n But since AB is parallel to DE, and CD is parallel to FA, EF is parallel to BC. So, the lines AB, CD, EF are three lines that are not parallel to each other, hence forming a triangle. \n\n Now, the sides of this triangle are given as 200, 240, 300. Let me denote the lengths of the sides of triangle PQR as PQ = 200, QR = 240, RP = 300. Wait, but the problem doesn't specify which side corresponds to which, but since the triangle is arbitrary, maybe the side length of the hexagon can be found regardless of the order. \n\n Now, how does this triangle relate to the hexagon? Since the hexagon is equilateral and opposite sides are parallel, the triangle PQR is related to the hexagon's side length. \n\n I recall that in such hexagons, the triangle formed by the extensions of the sides AB, CD, EF is called the "outer triangle" of the hexagon. There's a formula that relates the side length of the hexagon to the sides of this outer triangle. \n\n After some research in my mind, I recall that in a convex equilateral hexagon with opposite sides parallel, the side length $ s $ of the hexagon is given by the formula: \n\n $$ \n \frac{1}{s} = \frac{1}{a} + \frac{1}{b} + \frac{1}{c} \n $$ \n\n where $ a, b, c $ are the sides of the outer triangle. Wait, but this formula seems similar to the reciprocal relation. Let me check with the given triangle sides. \n\n If the triangle sides are 200, 240, 300, then: \n\n $$ \n \frac{1}{s} = \frac{1}{200} + \frac{1}{240} + \frac{1}{300} \n $$ \n\n Calculating this: \n\n First, find a common denominator. The least common multiple of 200, 240, 300 is 1200. \n\n $$ \n \frac{1}{200} = \frac{6}{1200}, \quad \frac{1}{240} = \frac{5}{1200}, \quad \frac{1}{300} = \frac{4}{1200} \n $$ \n\n Adding them up: \n\n $$ \n \frac{6 + 5 + 4}{1200} = \frac{15}{1200} = \frac{1}{80} \n $$ \n\n Therefore, \n\n $$ \n \frac{1}{s} = \frac{1}{80} \implies s = 80 \n $$ \n\n Wait, that seems straightforward. But is this formula correct? Let me verify. \n\n Alternatively, think of the triangle formed by the extensions of AB, CD, EF as being related to the hexagon through some reciprocal relation. Since the hexagon is equilateral and the opposite sides are parallel, the triangle sides are related to the side length of the hexagon through the reciprocals. \n\n Alternatively, think of the triangle as being formed by three lines, each of which is parallel to a side of the hexagon. The distance between these lines (the sides of the triangle) is related to the side length of the hexagon. However, since the triangle is not equilateral, the distances would vary, leading to the reciprocal relation. \n\n Alternatively, think of the hexagon as being formed by the triangle PQR with three lines cutting it. Each side of the hexagon is a segment between two parallel lines (opposite sides). The length of the hexagon's side would be related to the distances between these lines and the angles between them. \n\n But since the triangle sides are given, and the formula gives s = 80, which is a clean answer, maybe that's the correct answer. \n\n Alternatively, let me think of an example. If the triangle sides are 200, 240, 300, then the side length of the hexagon is 80. Let me check if this makes sense. \n\n Alternatively, think of the triangle sides as being the lengths of the projections of the hexagon's sides onto the directions of the triangle's sides. But since the hexagon is equilateral and opposite sides are parallel, maybe the sides of the triangle are related to the side length of the hexagon through some linear combination. \n\n Alternatively, think of the triangle formed by the extensions of AB, CD, EF as similar to the triangle formed by the sides of the hexagon. But since the hexagon is equilateral, that triangle would be equilateral. However, the given triangle is not equilateral, so that might not hold. \n\n But according to the formula I derived earlier, using the reciprocals, the answer is 80. Let me check if that makes sense. \n\n Alternatively, think of the hexagon as being formed by cutting the triangle PQR with three parallel lines. Each pair of opposite sides of the hexagon is parallel to a side of the triangle. The distance between these lines would be related to the side length of the hexagon. \n\n But since the triangle sides are 200, 240, 300, maybe the side length of the hexagon is the harmonic mean of these sides. The harmonic mean of 200, 240, 300 is: \n\n $$ \n \frac{3}{\frac{1}{200} + \frac{1}{240} + \frac{1}{300}} = \frac{3}{\frac{15}{1200}} = \frac{3600}{15} = 240 \n $$ \n\n Wait, but this contradicts the earlier result. Which one is correct? \n\n Alternatively, maybe I confused the formula. Let me think again. \n\n In the case of a regular hexagon, if we extend its sides, the triangle formed would be an equilateral triangle. However, in our case, the triangle is not equilateral. So, the formula must be different. \n\n Alternatively, think of the triangle formed by the extensions of AB, CD, EF as being related to the side length of the hexagon through the following: each side of the triangle is the sum of the side length of the hexagon and some multiple of the side length. But I need to think more carefully. \n\n Alternatively, think of the triangle formed by the extensions of AB, CD, EF as a triangle where each side is equal to the side length of the hexagon divided by some

contain AB, CD, EF? But AB and CD are parallel, so they can't form a triangle. Therefore, there must be a misunderstanding in the problem statement. Wait, let me check again. \n\n The problem says: "the triangle whose sides are extensions of segments AB, CD, and EF". So, the triangle has sides that are extensions of AB, CD, EF. So, each side of the triangle is an extension of one of these segments. Therefore, the triangle is formed by three lines, each of which is an extension of AB, CD, or EF. However, if AB and CD are parallel, then they can't form a triangle. Therefore, maybe the triangle is formed by extending AB, CD, and EF in such a way that they are not the lines AB, CD, EF, but their extensions beyond certain points? \n\n Alternatively, maybe the triangle is formed by extending AB, CD, and EF beyond their respective segments so that they intersect? \n\n Wait, let me think again about the parametrization. Let me take line AB: from A(0) to B(**a**). Line CD: from D(**a** + **b** + **c**) to E(**b** + **c**). The direction of CD is -**a**. Therefore, line CD is parallel to AB. Therefore, if I extend AB beyond B, and extend CD beyond E, they are still parallel. Similarly, if I extend AB beyond A and CD beyond D, they are still parallel. Therefore, AB and CD are parallel lines, so they can't form a triangle with EF. Therefore, maybe the triangle is formed by extending AB, CD, and EF in different directions? \n\n Alternatively, maybe the triangle is formed by extending AB, CD, and EF not in the order I thought. Maybe the triangle is formed by extending AB, EF, and FA? But the problem says AB, CD, EF. \n\n Wait, maybe the problem is not about the lines AB, CD, EF, but about the lines that are parallel to AB, CD, EF? But the problem says "extensions of segments AB, CD, and EF". So, it's the lines that contain AB, CD, EF. But since AB and CD are parallel, they can't form a triangle. Therefore, there must be a mistake in my previous analysis. \n\n Wait, maybe my mistake is in assuming that DE = -**a**? Let me check again. \n\n Earlier, I said that DE is parallel to AB, so DE = k**a**, and since |DE| = s, then k = ?1. Then, using the closure of the hexagon, we found that k = m = n = -1. However, maybe that is incorrect? \n\n Wait, the closure equation is AB + BC + CD + DE + EF + FA = 0. \n\n Substituting DE = k**a**, EF = m**b**, FA = n**c**. \n\n Therefore, **a** + **b** + **c** + k**a** + m**b** + n**c** = 0. \n\n Grouping terms: \n\n (1 + k)**a** + (1 + m)**b** + (1 + n)**c** = 0. \n\n Since **a**, **b**, **c** are not colinear (as the hexagon is convex), the coefficients must be zero. Therefore: \n\n 1 + k = 0 => k = -1, \n\n 1 + m = 0 => m = -1, \n\n 1 + n = 0 => n = -1. \n\n Therefore, that is correct. Therefore, DE = -**a**, EF = -**b**, FA = -**c**. Therefore, line CD is from D to E, direction -**a**. Therefore, line CD is parallel to AB. Therefore, AB and CD are parallel. Therefore, how can they form a triangle? \n\n Wait, perhaps the triangle is not formed by AB, CD, EF, but by AB, CD, and FA? But the problem says AB, CD, EF. \n\n Alternatively, maybe the triangle is formed by extending AB, CD, and EF in such a way that they are not the lines AB, CD, EF, but their extensions in different directions? \n\n Wait, maybe the triangle is formed by the three lines that are parallel to AB, CD, EF? But that would be three parallel lines, which can't form a triangle. \n\n Alternatively, maybe the triangle is formed by extending AB, CD, and EF beyond their segments so that they meet at different points. Wait, if AB is extended in one direction, CD in another, and EF in another, maybe they meet at three points forming a triangle. \n\n Let me think again. Let me consider the lines AB, CD, EF. Since AB and CD are parallel, they don't meet, but maybe if I extend AB and EF, they meet at a point, and CD and EF meet at another point, and AB and CD meet... but they don't. Wait, this is confusing. \n\n Alternatively, maybe the triangle is formed by the three lines that are parallel to AB, CD, EF? But that would be three parallel lines, which can't form a triangle. \n\n Alternatively, maybe the triangle is formed by the three lines that are not AB, CD, EF, but their extensions. For example, if AB is extended beyond B, CD is extended beyond D, and EF is extended beyond E, then maybe they meet at three points forming a triangle. \n\n Alternatively, maybe the triangle is formed by the three lines that are not the sides AB, CD, EF, but their extensions in such a way that they are not parallel. \n\n Wait, perhaps I need to think of the triangle as formed by the three lines that are the extensions of AB, CD, and EF, but in such a way that each line is extended in the direction opposite to the side. For example, AB is extended beyond A, CD is extended beyond D, and EF is extended beyond E. Then, maybe these three lines intersect to form a triangle. \n\n Alternatively, maybe the triangle is formed by the three lines that are the extensions of AB, CD, and EF in the direction opposite to the hexagon's sides. \n\n Alternatively, maybe the triangle is formed by the three lines that are the extensions of AB, CD, and EF beyond their respective endpoints. For example, AB is extended beyond B, CD is extended beyond D, EF is extended beyond E. Then, these three lines might intersect at three points forming a triangle. \n\n Let me try to find the intersections of these extended lines. \n\n First, Line AB extended beyond B: This is the line AB, which is parametrized as t**a**, t in ?. If we extend beyond B, that's t > 1. \n\n Line CD extended beyond D: Line CD is from D(**a** + **b** + **c**) to E(**b** + **c**), direction vector -**a**. So, parametrizing line CD as D + s(-**a**) = **a** + **b** + **c** - s**a**. Extending beyond D would be s < 0. \n\n Line EF extended beyond E: Line EF is from E(**b** + **c**) to F(**c**), direction vector -**b**. Parametrizing as E + r(-**b**) = **b** + **c** - r**b**, r in ?. Extending beyond E would be r > 1. \n\n Now, let's find the intersections of these three lines. \n\n First, intersection of AB extended beyond B (t**a**) and CD extended beyond D (**a** + **b** + **c** - s**a**): \n\n Set t**a** = **a** + **b** + **c** - s**a** \n\n => (t + s)**a** = **a** + **b** + **c** \n\n Which, as before, implies that **a** + **b** + **c** must be a scalar multiple of **a**, which is not possible unless **b** and **c** are zero. Therefore, no solution. So, these two lines don't intersect? But they are parallel? \n\n Wait, AB is direction **a**, CD is direction -**a**, so they are parallel. Therefore, they don't intersect. Therefore, this approach is not working. \n\n Alternatively, maybe the triangle is formed by extending AB beyond A, CD beyond D, and EF beyond E? Let's try that. \n\n Line AB extended beyond A: parametrized as a triangle. Alternatively, maybe extending AB beyond A, CD beyond C, EF beyond E? The exact way of extension is important. \n\n Alternatively, maybe the triangle is formed by the three lines that contain AB, CD, EF. Since AB is parallel to DE, CD is parallel to FA, EF is parallel to BC. So, the lines AB, CD, EF are three lines, not necessarily parallel, forming a triangle when extended. \n\n Let me consider the triangle formed by the lines AB, CD, EF. Let me denote the intersections of these lines. Let me suppose that line AB and line CD intersect at point P, line CD and line EF intersect at point Q, and line EF and line AB intersect at point R. Then triangle PQR is the triangle in question, with sides 200, 240, 300. The problem says that this triangle has sides 200, 240, 300. The question is to find the side length s of the hexagon. \n\n So, the triangle PQR has sides PQ = 200, QR = 240, RP = 300, or some permutation. But since the problem says "the triangle whose sides are extensions of segments AB, CD, and EF", maybe the sides of the triangle are the lengths of the intersections? Wait, maybe not. The sides of the triangle are the lengths between the intersections. So, for example, PQ is the length between the intersection of AB and CD, QR is the length between the intersection of CD and EF, and RP is the length between the intersection of EF and AB. \n\n Alternatively, maybe the triangle is formed by extending AB, CD, EF in such a way that each side of the triangle is a segment formed by the extensions. However, without a diagram, it's a bit challenging, but perhaps we can use the properties of the hexagon to relate the triangle sides to the hexagon side length. \n\n Alternatively, maybe using the concept of similar triangles. Since the hexagon has opposite sides parallel, the triangle formed by extending AB, CD, EF might be similar to some triangle related to the hexagon. \n\n Alternatively, think of the hexagon as part of the triangle. If we consider the triangle PQR, then the hexagon is inscribed in this triangle in some way. Maybe the sides of the hexagon are related to the sides of the triangle through some ratio. \n\n Alternatively, since the hexagon is equilateral and has opposite sides parallel, maybe there is a formula that relates the side length of the hexagon to the sides of the triangle. \n\n Alternatively, think about the fact that in such a hexagon, the distances between the opposite sides are proportional to the sides of the triangle. But I need to think of a way to relate these distances. \n\n Alternatively, think of the triangle PQR as a triangle with three lines AB, CD, EF, and the hexagon lies between these lines. Since AB is parallel to DE, and DE is a side of the hexagon, which is opposite to AB. Similarly for the other sides. \n\n Alternatively, think of the hexagon as being formed by three pairs of parallel lines. Each pair of opposite sides is a line. The hexagon is between these lines. The triangle formed by the extensions of AB, CD, EF is the outer triangle, and the hexagon is inside. \n\n Alternatively, think of the triangle as being formed by three lines, each of which is a side of the hexagon extended. The distances between these lines would be related to the side length of the hexagon. \n\n Alternatively, maybe using area ratios or something else. \n\n Alternatively, think of the triangle PQR and the hexagon as related through homothety. If there is a homothety (a scaling transformation) that maps the triangle to the hexagon or vice versa. \n\n Alternatively, since the hexagon is equilateral with opposite sides parallel, it can be considered as a centrally symmetric hexagon. Therefore, the triangle formed by the extensions might be related to the hexagon through some central symmetry. \n\n Alternatively, think of coordinates again. Maybe assign coordinates to the triangle and then find coordinates of the hexagon. \n\n Let me try this approach. Let me suppose that triangle PQR has sides of length 200, 240, 300. Let me assign coordinates to this triangle. Let me place triangle PQR in coordinate system such that point P is at (0, 0), point Q is at (200, 0), and point R is somewhere in the plane. Wait, but the triangle has sides 200, 240, 300. Let me check if it's a valid triangle. The sum of the two smaller sides: 200 + 240 = 440 > 300, so yes, valid. \n\n But to make it easier, maybe assign coordinates such that the triangle has coordinates that can be easily worked with. Alternatively, use vectors. \n\n Alternatively, since the hexagon is equilateral and opposite sides are parallel, maybe the triangle PQR is similar to the triangle formed by the vectors AB, BC, CD. \n\n Wait, earlier we saw that AB + BC + CD = 0. So, vectors AB, BC, CD form a triangle. Wait, but AB + BC + CD = 0 implies that AB + BC + CD = 0, so if you start at point A, go along AB, then BC, then CD, you end up back at A? No, because AB + BC + CD = 0, so starting at A, going AB to B, then BC to C, then CD to D, then from D, the vector DA is - (AB + BC + CD) = 0, so D = A? Which is not possible. Wait, that seems contradictory. Wait, but earlier in the regular hexagon, AB + BC + CD was not zero. Therefore, my mistake must be in the assumption that AB + BC + CD = 0. Wait, earlier we had that AB + BC + CD + DE + EF + FA = 0, and since DE = -AB, EF = -BC, FA = -CD, then substituting gives AB + BC + CD - AB - BC - CD = 0, which is correct. However, AB + BC + CD is not necessarily zero. Therefore, the confusion arises here. So, the sum AB + BC + CD is equal to -(DE + EF + FA). But DE + EF + FA is the sum of the other three sides, which are negatives of AB, BC, CD. So, AB + BC + CD = -(DE + EF + FA). But since DE = -AB, EF = -BC, FA = -CD, then DE + EF + FA = -AB - BC - CD, so AB + BC + CD = - (-AB - BC - CD) = AB + BC + CD. Which is trivially true. Therefore, this doesn't give us any new information. \n\n Therefore, perhaps I need another approach. \n\n Let me think about the triangle formed by the extensions of AB, CD, EF. Let me denote this triangle as triangle XYZ, where X is the intersection of AB and CD, Y is the intersection of CD and EF, Z is the intersection of EF and AB. The sides of triangle XYZ are given as 200, 240, 300. The problem is to find the side length of the hexagon. \n\n Now, since AB is parallel to DE, and CD is parallel to FA, and EF is parallel to BC, then maybe the triangle XYZ is similar to some triangle related to the hexagon. \n\n Alternatively, since AB is parallel to DE, and DE is a side of the hexagon, then the distance between AB and DE is constant. Similarly for the other pairs of sides. \n\n Alternatively, think of the hexagon as being formed by three pairs of parallel lines, with AB || DE, BC || EF, CD || FA. The triangle XYZ is formed by extending AB, CD, EF. \n\n Let me consider the lines AB, CD, EF. Since AB is parallel to DE, factor. But without more information, this is too vague. \n\n Alternatively, recall that in a convex equilateral hexagon with opposite sides parallel, the triangle formed by the extensions of AB, CD, EF is called the "outer triangle" and the side length of the hexagon is given by the reciprocal of the sum of the reciprocals of the sides of the outer triangle. \n\n Wait, if that is the case, then: \n\n $$ \n \frac{1}{s} = \frac{1}{200} + \frac{1}{240} + \frac{1}{300} \n $$ \n\n Which gives: \n\n $$ \n \frac{1}{s} = \frac{6 + 5 + 4}{1200} = \frac{15}{1200} = \frac{1}{80} \n $$ \n\n Therefore, $ s = 80 $. \n\n This seems to be a plausible answer, and the calculation checks out. Therefore, the side length of the hexagon is 80. \n\n **Final Answer** \n \boxed{80} \n </think> \n\n To determine the side length of the convex equilateral hexagon $ ABCDEF $, we are told that: \n\n - All sides of the hexagon are equal in length, denoted as $ s $. \n - All pairs of opposite sides are parallel. \n - A triangle is formed by extending the sides $ AB $, $ CD $, and $ EF $, and this triangle has side lengths of 200, 240, and 300. \n\n --- \n\n ### Key Insight \n\n This type of hexagon is a special case of an **equilateral hexagon with opposite sides parallel**. In such configurations, the triangle formed by extending the sides $ AB $, $ CD $, and $ EF $ is known as the **outer triangle** of the hexagon. \n\n There is a known geometric relationship between the side length $ s $ of the hexagon and the side lengths $ a, b, c $ of the outer triangle: \n\n $$ \n \frac{1}{s} = \frac{1}{a} + \frac{1}{b} + \frac{1}{c} \n $$ \n\n This relationship arises from the fact that the triangle is formed by lines that are parallel to the sides of the hexagon, and the side length of the hexagon is related to the **reciprocals** of the triangle's side lengths. \n\n --- \n\n ### Applying the Formula \n\n Given that the triangle has side lengths 200, 240, and 300, we apply the formula: \n\n $$ \n \frac{1}{s} = \frac{1}{200} + \frac{1}{240} + \frac{1}{300} \n $$ \n\n To compute the sum: \n\n - The least common denominator (LCD) of 200, 240, and 300 is 1200. \n - Convert each fraction: \n\n $$ \n \frac{1}{200} = \frac{6}{1200}, \quad \frac{1}{240} = \frac{5}{1200}, \quad \frac{1}{300} = \frac{4}{1200} \n $$ \n\n Add them: \n\n $$ \n \frac{6 + 5 + 4}{1200} = \frac{15}{1200} = \frac{1}{80} \n $$ \n\n So: \n\n $$ \n \frac{1}{s} = \frac{1}{80} \Rightarrow s = 80 \n $$ \n\n --- \n\n ### Final Answer \n\n $$ \n \boxed{80} \n $$

t**a**, t < 0. \n\n Line CD extended beyond D: same as before, **a** + **b** + **c** - s**a**, s in ?. \n\n Line EF extended beyond E: **b** + **c** - r**b**, r > 1. \n\n Again, trying to find intersections. \n\n Intersection of AB extended beyond A (t**a**) and CD extended beyond D (**a** + **b** + **c** - s**a**): \n\n t**a** = **a** + **b** + **c** - s**a** \n\n => (t + s)**a** = **a** + **b** + **c** \n\n Same equation as before. Still no solution. So, again, no intersection. \n\n Alternatively, maybe the triangle is formed by extending AB beyond B, CD beyond E, and EF beyond F? \n\n Let me try that. \n\n Line AB extended beyond B: t**a**, t > 1. \n\n Line CD extended beyond E: Line CD is from D to E, direction -**a**. Extending beyond E would be E + s(-**a**) = **b** + **c** - s**a**. \n\n Line EF extended beyond F: Line EF is from E to F, direction -**b**. Extending beyond F would be F + r(-**b**) = **c** - r**b**. \n\n Now, let's find intersections. \n\n First, intersection of AB extended beyond B (t**a**) and CD extended beyond E (**b** + **c** - s**a**): \n\n t**a** = **b** + **c** - s**a** \n\n => (t + s)**a** = **b** + **c** \n\n Similarly, this equation implies that **b** + **c** is a scalar multiple of **a**, which is not true unless **b** and **c** are colinear with **a**, which they are not in a general hexagon. Therefore, no solution. \n\n Hmm, this is getting frustrating. Maybe my entire approach is wrong. Let me try to think differently. \n\n The problem says that the triangle formed by the extensions of AB, CD, and EF has sides 200, 240, 300. Maybe instead of trying to find coordinates, I can use some properties of equilateral hexagons with opposite sides parallel. \n\n I recall that in such hexagons, there's a relation between the sides of the triangle and the side length of the hexagon. Maybe similar to the formula for a parallelogram or something else. \n\n Alternatively, think of the hexagon as a combination of three parallelograms. Since opposite sides are parallel, each pair of opposite sides can be considered as a pair of parallel lines. \n\n Alternatively, think of the hexagon as being formed by a triangle with three pairs of parallel lines. Wait, maybe the triangle mentioned in the problem is related to the hexagon in such a way that the sides of the triangle are related to the distances between the sides of the hexagon. \n\n Alternatively, consider that the hexagon is affine equivalent to a regular hexagon. In a regular hexagon, the opposite sides are parallel and equal. However, in this case, the hexagon is equilateral but not necessarily regular. However, the triangle formed by extending the sides might be similar to the triangle formed by extending the sides of a regular hexagon. \n\n Alternatively, think of the triangle formed by the extensions of AB, CD, EF as the outer triangle of the hexagon. In a regular hexagon, extending the sides would form a larger triangle, but since this hexagon is not regular, the triangle has sides 200, 240, 300. \n\n Alternatively, maybe the triangle is similar to the triangle formed by the sides of the hexagon, scaled by some factor. \n\n Alternatively, think of the hexagon as being in a coordinate system where the triangle is given, and the hexagon is inscribed in it. \n\n Alternatively, use the concept of homothety. If the triangle is formed by extending the sides of the hexagon, then there might be a homothety (scaling) that relates the hexagon to the triangle. \n\n Alternatively, consider that the sides of the triangle are related to the side length of the hexagon through some trigonometric relations. \n\n Alternatively, think of the hexagon as a central triangle with three pairs of parallel sides. Wait, maybe not. \n\n Alternatively, think of the triangle formed by the extensions of AB, CD, EF as a triangle where each side is parallel to a side of the hexagon. Wait, but AB is parallel to DE, CD is parallel to FA, EF is parallel to BC. So, the triangle has sides that are parallel to AB, CD, EF, which are each parallel to DE, FA, BC respectively. Therefore, the triangle has sides parallel to three different directions, which are not necessarily related. \n\n Alternatively, think of the triangle as having sides that are related to the sides of the hexagon through some vector relations. \n\n Alternatively, think about the fact that in such a hexagon, the sides of the triangle are related to the side length 's' and the angles between the sides. \n\n Alternatively, maybe use the area of the triangle and the area of the hexagon, but I don't see the connection. \n\n Alternatively, think of the triangle formed by the extensions as a triangle where each side is a translation of the corresponding side of the hexagon. \n\n Alternatively, maybe use the concept of reciprocal vectors or something. \n\n Alternatively, let me think of the triangle formed by the extensions. Let me assume that the triangle has sides 200, 240, 300. Let me denote them as a = 200, b = 240, c = 300. \n\n Now, in the hexagon, each side is length s. The triangle is formed by extending AB, CD, EF. Let me try to relate the sides of the triangle to the side length s. \n\n Maybe the triangle is similar to a triangle formed by the vectors **a**, **b**, **c**. But and DE is another side of the hexagon. Similarly, CD is parallel to FA, and EF is parallel to BC. \n\n Now, if I can find the relations between the distances between these lines and the side length of the hexagon, maybe I can find the side length. \n\n Alternatively, think of the triangle XYZ and the hexagon as being related by the side length s. Maybe the sides of triangle XYZ are proportional to s. \n\n Alternatively, think of the triangle XYZ as being formed by three lines, each of which is a side of the hexagon extended. The hexagon lies between these lines. Since opposite sides are parallel, the distances between AB and DE, BC and EF, CD and FA are related to the side length s. \n\n Alternatively, think of the hexagon as a central hexagon within triangle XYZ. Maybe the side length s is related to the sides of XYZ through some formula. \n\n Alternatively, think of the hexagon as being formed by cutting off three small triangles from triangle XYZ. If that's the case, then the side length of the hexagon could be related to the sides of XYZ. \n\n Alternatively, think of the triangle XYZ and the hexagon as being related by similar triangles. For example, if the hexagon is similar to triangle XYZ, but that might not be the case. \n\n Alternatively, since the hexagon is equilateral and opposite sides are parallel, maybe the triangle XYZ is similar to the triangle formed by the vectors AB, BC, CD. Wait, earlier we saw that AB + BC + CD = 0, so vectors AB, BC, CD form a triangle. Therefore, the triangle formed by AB, BC, CD is a triangle with sides equal to the lengths of AB, BC, CD, which are all equal to s. Wait, but AB, BC, CD are three vectors that form a triangle, but they are not necessarily of the same length? Wait, no, in our case, the hexagon is equilateral, so AB = BC = CD = s. Therefore, the triangle formed by vectors AB, BC, CD is an equilateral triangle? Wait, but AB + BC + CD = 0, so the vectors AB, BC, CD form a triangle. If AB, BC, CD are all of length s, then this triangle is equilateral. However, in reality, in a regular hexagon, the vectors AB, BC, CD do not form an equilateral triangle. Wait, in the regular hexagon example earlier, AB + BC + CD was equal to vector AD, which was (-2, 0), but AB, BC, CD are vectors of length 1, but their sum is not zero. So, this seems contradictory. Therefore, perhaps my assumption that AB, BC, CD form a triangle is not correct. \n\n Wait, no, in the regular hexagon, AB + BC + CD is not zero, but in our general case, AB + BC + CD = 0. Wait, no, earlier we saw that AB + BC + CD + DE + EF + FA = 0, and since DE = -AB, EF = -BC, FA = -CD, then AB + BC + CD - AB - BC - CD = 0. Therefore, this is always true. But in the regular hexagon, AB + BC + CD is not zero. Wait, this is confusing. There must be a mistake in the regular hexagon example. \n\n Wait, in the regular hexagon, AB is a vector, BC is another vector, CD is another vector. If AB + BC + CD = 0, then in the regular hexagon, that would mean that vector AD = 0, which is not true. Therefore, there is a contradiction here. Therefore, my previous conclusion must be wrong. Where is the mistake? \n\n Wait, earlier I said that DE = - AB, EF = - BC, FA = - CD. But in the regular hexagon, DE is from D to E, which is vector DE = E - D. In the regular hexagon with coordinates as before, D is (-1, 0), E is (-0.5, -sqrt3/2). So vector DE is E - D = (-0.5 - (-1), -sqrt3/2 - 0) = (0.5, -sqrt3/2). Vector AB is B - A = (0.5 - 1, sqrt3/2 - 0) = (-0.5, sqrt3/2). So DE = (0.5, -sqrt3/2) = - (-0.5, sqrt3/2) = - AB. Yes, so DE = - AB. Similarly, EF = - BC, FA = - CD. Therefore, in the regular hexagon, AB + BC + CD + DE + EF + FA = AB + BC + CD - AB - BC - CD = 0. So that's correct. But in the regular hexagon, AB + BC + CD is not zero. However, AB + BC + CD + DE + EF + FA = 0, so AB + BC + CD = - (DE + EF + FA). But DE + EF + FA = - AB - BC - CD. Therefore, AB + BC + CD = AB + BC + CD. So, this is trivially true. Therefore, there is no contradiction. The confusion arises because in the regular hexagon, AB + BC + CD is not zero, but AB + BC + CD = - (DE + EF + FA), which is also not zero. Therefore, the previous conclusion that AB + BC + CD = 0 is wrong. Wait, no, the sum AB + BC + CD + DE + EF + FA = 0, and DE + EF + FA = - (AB + BC + CD). Therefore, AB + BC + CD + (- (AB + BC + CD)) = 0. Which is correct. Therefore, there is no contradiction. The confusion was in thinking that AB + BC + CD = 0, but actually, it is not. Therefore, my previous mistake was in thinking that AB + BC + CD = 0. It is not. The correct equation is AB + BC + CD = - (DE + EF + FA). Therefore, this doesn't give us direct information about AB + BC + CD. \n\n So, returning to the problem. Maybe I need to think of the triangle XYZ formed by the extensions of AB, CD, EF. Let me try to find relations between the sides of this triangle and the side length s of the hexagon. \n\n Let me consider the triangle XYZ. Let me denote the intersection of AB and CD as X, CD and EF as Y, and EF and