

STIR: Self-Distilled Latent Tools for Dynamic Control of LLM Reasoning

Zhenning Shi^{1*}, Yijia Zhu^{1*}, Junhan Shi¹, Xun Zhang², Lei Wang¹, Congcong Miao^{3†}

¹Tsinghua University, Beijing, China

²Fuzhou University, Fuzhou, China

³National University of Singapore, Singapore

{shizn23, shijh23, wanglei23}@tsinghua.org.cn,

zyg26@mails.tsinghua.edu.cn, 102304229@fzu.edu.cn, mccmiao@163.com

Abstract

Chain-of-thought reasoning improves large language models by spending more tokens at test time, but this couples accuracy to generation length. Activation steering offers a lightweight alternative, yet most steering methods apply a fixed vector throughout generation and cannot adapt to different stages of a multi-step solution. We propose STIR (Self-Distilled Tools for Internal Reasoning), a test-time framework that builds a compact library of state-conditioned correction vectors from the model’s own successful and failed rollouts. At inference time, STIR retrieves candidate corrections at structural reasoning checkpoints, validates them with a short lookahead probe, and abstains when the current trajectory already matches a high-quality state. Across six mathematical and knowledge-intensive reasoning benchmarks, STIR improves average accuracy by 1.9% to 7.5% over Vanilla decoding while reducing average token consumption by up to 35%. These results suggest that selective latent interventions can improve the accuracy-cost trade-off of reasoning without forcing every problem to generate longer chains. Our code is available at <https://github.com/sznnzs/STIR>.

1 Introduction

Large language models increasingly solve hard problems by spending more computation at test time (Zhang et al., 2025). Chain-of-thought prompting serializes intermediate reasoning into explicit tokens (Wei et al., 2022; Kojima et al., 2022; Zhou et al., 2023), while tree and graph search, sampling, and voting methods improve reliability by exploring multiple candidate paths (Yao et al., 2023; Besta et al., 2024; Snell et al., 2025). This token-centric strategy (Xiao et al., 2025a,b, 2026a) is effective, but it ties stronger reasoning to longer or repeated generations. In practical deployment,

additional deliberation often means higher latency, higher serving cost, and a larger risk of verbose or repetitive reasoning (Zeng et al., 2025; Sui et al., 2025; Xiao et al., 2026c).

A complementary direction is to improve reasoning inside the model’s hidden states. Work on implicit reasoning and latent computation suggests that some intermediate computation can be represented without exposing every step as text (Hao et al., 2025; Zhu et al., 2025; Shen et al., 2025). However, many such methods allocate special latent, pause, or internal tokens, and often require additional training or careful optimization to keep the hidden trajectory stable (Zelikman et al., 2024; Skean et al., 2025; Lin et al., 2025; Kim et al., 2025; London and Kanade, 2025). These requirements make them hard to use as lightweight repair mechanisms for an already pretrained model. We therefore ask whether a model’s reasoning trajectory can be improved at test time, without retraining the model or forcing every instance to produce longer explicit chains.

Activation steering offers a natural mechanism for this goal because it intervenes directly in the residual stream without modifying model parameters (Rimsky et al., 2024; Turner et al., 2023). Early steering work focused on global attributes through function vectors or representation engineering (Todd et al., 2024), and recent work has begun to apply steering to reasoning behaviors such as overthinking or repetitive continuation (Chen et al., 2025b; Huang et al., 2025; Azizi et al., 2026). Despite this promise, most steering protocols still rely on static control vectors that impose one global bias across generation. This static paradigm is poorly matched to multi-step reasoning: a correction that helps decomposition may differ from one that helps verification, and an intervention that helps one phase can disturb another. Reasoning control therefore needs to be dynamic, state-aware (Xiao et al., 2026b), and able to abstain.

*Equal contribution.

†Corresponding author.

To address these challenges, we propose STIR (Self-Distilled Tools for Internal Reasoning), a test-time framework for state-conditioned activation steering. The key idea is to mine useful control signals from the model’s own exploration space. Even when greedy decoding fails, stochastic rollouts may contain correct subpaths or complete successful solutions. STIR compares successful and failed local states at structural reasoning checkpoints, extracts their hidden-state differences as correction vectors, and stores them as reusable latent tools. The library also includes null-action anchors for states where intervention is unnecessary.

STIR has three stages. First, *Contrastive Tool Induction* aligns stochastic rollouts at reasoning checkpoints and computes local contrasts between successful and failed prefixes. Second, *Sparse Tool Library Construction* filters the raw vectors into a compact and diverse library, reducing redundant corrections during online retrieval. Finally, *Lookahead-Gated Intervention* retrieves candidate tools for the current hidden state, validates them with a short lookahead probe, and applies a bounded correction only when local evidence supports intervention. If the current state already resembles a successful prefix, the controller selects the null action and leaves the trajectory unchanged.

We evaluate STIR on six mathematical and knowledge-intensive reasoning benchmarks across four main open-weight model settings. STIR improves average accuracy by 1.9% to 7.5% over Vanilla decoding while reducing average token cost by up to 35%. Ablations and diagnostic analyses clarify which components drive these gains and how the controller behaves across tasks.

Our contributions are as follows:

- We identify the mismatch between static activation steering and state-dependent reasoning, motivating selective interventions at structural reasoning checkpoints.
- We introduce STIR, which mines correction vectors from successful and failed rollouts, organizes them into a compact latent-tool library, and pairs active corrections with null-action anchors for principled abstention.
- We design an inference-time controller that retrieves, probes, and conditionally applies latent tools, allowing the model to preserve good trajectories while correcting uncertain ones.
- We provide an empirical study across six benchmarks and four main model settings, showing improved average accuracy-token trade-offs and ablations that explain where the gains come from.

2 Related Work

Reasoning with Large Language Models. Current strategies for improving large language model reasoning primarily rely on explicit token generation (Zelikman et al., 2022). Chain-of-thought prompting (Wei et al., 2022) shows that eliciting intermediate steps can substantially improve performance on complex tasks (Kojima et al., 2022). Later methods organize generation into richer structures, including tree-based search (Yao et al., 2023) and graph-based reasoning (Besta et al., 2024), which support backtracking and lookahead. Sampling and voting methods further improve reliability by aggregating diverse stochastic trajectories (Snell et al., 2025). These approaches are effective, but they tie stronger reasoning to additional generated tokens and leave each sampled hidden-state trajectory largely unchanged. STIR complements this line of work by improving a single hidden trajectory rather than relying only on longer or repeated textual deliberation.

Implicit Reasoning. Recent research investigates how much reasoning can be internalized in hidden states without explicit token output (Hao et al., 2025; Zhu et al., 2025; Shen et al., 2025). Studies on implicit chain-of-thought suggest that transformer models can perform nontrivial deductions internally (Zelikman et al., 2024; Skea et al., 2025), compressing parts of the reasoning process into the residual stream (Lin et al., 2025). Related work on pause tokens and internal computation tokens allocates additional latent capacity by extending the sequence with non-semantic placeholders (Kim et al., 2025; London and Kanade, 2025). These methods show that latent computation is useful, but their internal process is difficult to route toward a specific local failure and may change the training or inference interface. STIR takes a complementary approach: it extracts reusable correction operators from observed success-failure contrasts and applies them selectively at structural checkpoints rather than throughout generation.

Latent Representation Steering. Mechanistic interpretability has enabled direct intervention in a

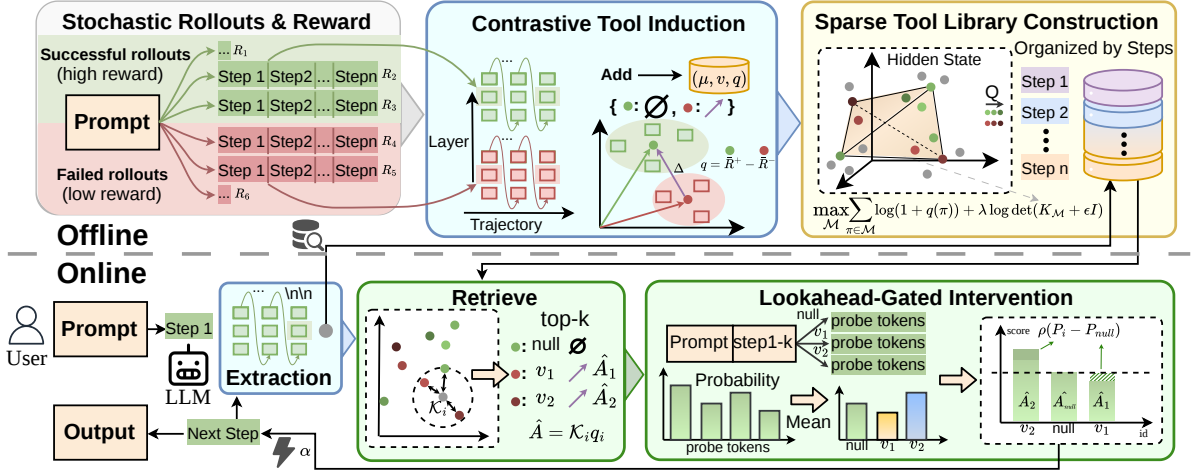


Figure 1: Overview of STIR. The offline stage samples stochastic rollouts, aligns prefixes at structural reasoning checkpoints, and extracts correction vectors from successful versus failed local states. A sparse library then stores both active corrections and null-action anchors. At inference time, STIR retrieves candidate tools for the current hidden state, validates them with a short lookahead probe, and applies a correction only when the local evidence supports intervention.

model’s internal representations (Todd et al., 2024). Activation steering (Rimsky et al., 2024) and representation engineering (Turner et al., 2023) show that high-level attributes can often be controlled by extracting and injecting directions in activation space (Stolfo et al., 2025; Azizi et al., 2026; Huang et al., 2025). Recent steering work further suggests that representation-level edits can affect reasoning length and calibration (Azizi et al., 2026; Huang et al., 2025). Most existing steering protocols use a fixed direction during inference, which is suitable for global attributes but less suitable for multi-step reasoning. Reasoning trajectories pass through heterogeneous phases such as retrieval, decomposition, deduction, and verification; a correction that helps one phase may be irrelevant or harmful in another. STIR extends representation steering from a static global edit to a state-conditioned tool library that decides which correction to apply and when to abstain through Lookahead-Gated Intervention.

3 Preliminaries and Problem Formulation

3.1 Latent Dynamics of Generative Reasoning

We analyze the generative reasoning of a transformer-based large language model $\mathcal{M}_\theta$ as a discrete dynamical system. Given an input context x , the model generates a reasoning chain $Y = (y_1, \dots, y_T)$ where the hidden state $h_t^{(l)}$ at timestep t and layer l evolves via residual dynamics:

$$h_t^{(l)} = h_t^{(l-1)} + \mathcal{F}_\theta^{(l)}(h_t^{(l-1)}, \text{Attn}(h_{<t})), \quad (1)$$

Here, the final state $h_t^{(L)}$ projects onto the vocabulary to determine the next-token probability $P(y_t | x, y_{<t}) = \text{softmax}(W_U h_t^{(L)})$. This residual structure implies that applying a targeted perturbation δ to $h_t^{(l)}$ can steer the generation trajectory without modifying the parameters θ .

3.2 The Challenge of Temporal Misalignment

Standard activation steering methods assume that a desired behavior corresponds to a global translation direction (Chen et al., 2025a; Huang et al., 2025), injecting a time-invariant vector v^* into the residual stream:

$$\tilde{h}_t \leftarrow h_t + \alpha \cdot v^*, \quad \forall t \in [1, T], \quad (2)$$

This static treatment is poorly matched to multi-step reasoning. Reasoning states change across phases such as planning, retrieval, deduction, and verification, so a direction that supports one phase may be unhelpful in another. We therefore study dynamic, state-dependent control, where interventions are selected according to the current reasoning checkpoint rather than applied uniformly across the full generation.

4 Methodology

STIR turns activation steering into a state-conditioned test-time controller. Rather than applying one global steering vector, it builds a library of local correction vectors from the model’s own rollouts and decides at each checkpoint whether a

correction is warranted. The framework has three stages. Contrastive Tool Induction extracts candidate vectors from successful and failed prefixes. Sparse Tool Library Construction selects a compact and diverse library of reusable tools. Lookahead-Gated Intervention retrieves candidate tools at inference time, validates them with a short lookahead probe, and applies a bounded correction only when the evidence supports intervention.

4.1 Contrastive Tool Induction

The offline stage uses the model’s own rollouts as supervision. For each mining prompt x , we sample K stochastic solutions and score each solution by correctness and length. This favors trajectories that reach the correct answer without unnecessary continuation, which is important because the mined vectors are later used to repair or shorten reasoning rather than to encourage longer deliberation.

$$R(Y) = \mathbb{I}(Y \in \mathcal{Y}^*) - \eta \cdot \frac{\mathcal{L}(Y)}{L_{max}}, \quad (3)$$

where $\mathbb{I}(\cdot)$ indicates correctness against ground truth, $\mathcal{L}(Y)$ denotes the generated length, and η controls the length penalty.

To extract latent steering impulses, we align sampled trajectories by step index at structural checkpoints. In our implementation, a checkpoint is created by the double-newline delimiter `\n\n`. We do not force rollouts with different lengths into a shared token-level alignment. Instead, each checkpoint index m has its own library bucket: only rollouts that reach step m contribute to the positive or negative set for that index, while earlier-terminating rollouts contribute nothing to later indices. During inference, retrieval is also conditioned on the checkpoint index, so tools mined from one step are not mixed with tools from another.

At a decision point m , we partition the reached rollout prefixes into a high-reward positive set $\mathcal{P}_m$ and a low-reward negative set $\mathcal{N}_m$. Under the hypothesis that the transition from failure to success can be locally approximated by a linear translation (Park et al., 2024), we compute centroid hidden states for the two sets:

$$\begin{aligned} \mu_m^+ &= \frac{1}{|\mathcal{P}_m|} \sum_{y \in \mathcal{P}_m} h_m(y), \\ \mu_m^- &= \frac{1}{|\mathcal{N}_m|} \sum_{y \in \mathcal{N}_m} h_m(y), \end{aligned} \quad (4)$$

The candidate steering impulse is the difference $v_m = \mu_m^+ - \mu_m^-$. We use this vector as a local correction proposal: it points from common failed prefixes toward nearby successful prefixes at the same checkpoint, without assuming that one global direction works for every reasoning state.

We also store evidence for when not to intervene. Each discovered contrast creates two entries. The correction entry is keyed by the negative centroid $k^- = \mu_m^-$ and stores the active impulse v_m . The anchor entry is keyed by the positive centroid $k^+ = \mu_m^+$ and stores a null impulse. During inference, a strong match to an anchor indicates that the current trajectory already resembles a successful prefix, allowing the controller to abstain instead of perturbing a valid solution.

4.2 Sparse Tool Library Construction

Contrastive Tool Induction can produce many similar candidate entries. Using all of them would slow retrieval and make the controller repeatedly consider near-duplicate corrections. We therefore select a compact library $\mathcal{M} \subset \Omega$ with budget $|\mathcal{M}| \leq B$. The selection objective preserves individually useful tools while discouraging redundant state keys.

Let $K \in \mathbb{R}^{N \times N}$ be the similarity kernel, where $K_{ij} = \langle k_i, k_j \rangle$ is the cosine alignment between state keys, and let $q(\pi)$ be the quality score of a tool estimated from reward gain during mining. We select $\mathcal{M}$ by maximizing

$$\begin{aligned} \mathcal{J}(\mathcal{M}) &= \sum_{\pi \in \mathcal{M}} \log(1 + q(\pi)) \\ &\quad + \lambda \log \det(K_{\mathcal{M}} + \epsilon I), \end{aligned} \quad (5)$$

where λ controls the diversity term and ϵI ensures numerical stability. The first term favors tools with higher observed utility, while the determinant term improves coverage by penalizing highly aligned keys.

Because exact maximization is NP-hard, we use the standard marginal-gain approximation for determinantal objectives (Gillenwater et al., 2012). At each iteration, the next entry is selected by its marginal gain:

$$\begin{aligned} \pi^* &= \arg \max_{\pi \in \Omega \setminus \mathcal{M}_{t-1}} \left[\log(1 + q(\pi)) \right. \\ &\quad \left. + \lambda \log \sigma_{\pi | \mathcal{M}_{t-1}}^2 \right], \end{aligned} \quad (6)$$

Here, $\sigma_{\pi | \mathcal{M}_{t-1}}^2$ measures how much new coverage the candidate key adds beyond the current library.

After selection, all keys are L2-normalized so on-line retrieval uses cosine similarity consistently across layers.

4.3 Lookahead-Gated Intervention

At inference time, STIR treats each checkpoint as a routing decision. Given the current hidden state h_t , it retrieves a small candidate set $\mathcal{C}_t$ from $\mathcal{M}$. Each tool π receives a retrieval prior $\hat{A}(\pi) = \mathcal{K}(h_t, k_\pi) \cdot q(\pi)$, where $\mathcal{K}$ is cosine similarity and $q(\pi)$ is the historical reward gain. If the strongest match is an anchor entry rather than an active correction, the controller prefers the null action and leaves the trajectory unchanged.

Similarity alone is not reliable enough in high-dimensional hidden states, so STIR validates the top candidates with a short lookahead probe. For each candidate, we run a lightweight T_{probe} token continuation and compare its confidence against the unperturbed continuation:

$$\mathcal{G}(\pi) = \frac{1}{T_{probe}} \sum_{j=1}^{T_{probe}} \left[\log P_\theta(\tilde{y}_{t+j} \mid \tilde{h}_\pi) - \log P_\theta(y_{t+j} \mid h_{default}) \right], \quad (7)$$

The probe filters out candidates that are close in representation space but unhelpful for the current continuation. STIR combines retrieval and probe evidence as $S(\pi) = \beta \cdot \hat{A}(\pi) + \rho \cdot \mathcal{G}(\pi)$, selects the best candidate π^* , and abstains if $S(\pi^*) < \tau_{null}$. If an active tool is selected, its injection strength is scaled by the resulting confidence:

$$h_t \leftarrow h_t + \text{clip}(k_{scale} \cdot S(\pi^*), 0, \alpha_{max}) \cdot v_{\pi^*}. \quad (8)$$

Thus, STIR intervenes only when retrieval and lookahead agree, and the perturbation magnitude remains bounded by α_{max} .

5 Experiments

5.1 Experimental Setup

Datasets and Target Models. We evaluate reasoning across six benchmarks. The mathematical benchmarks are AIME24, AIME25, AMC23 (math-ai, 2023), and MATH500 (Hendrycks et al., 2021); the knowledge-intensive benchmarks are ARC-Challenge (Clark et al., 2018) and OpenBookQA (OBQA) (Mihaylov et al., 2018). The main evaluation uses four open-weight model settings:

DeepSeek-R1-Distill-Qwen-1.5B, DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025), Qwen2.5-3B-Instruct, and Qwen2.5-7B-Instruct (Yang et al., 2025). These settings compare post-training style and scale within Qwen-derived models; Appendix B.3 adds Llama-3.1-8B-Instruct as a model-family check.

Baselines. We benchmark STIR against representative inference paradigms. Vanilla decoding measures the intrinsic single-trajectory capability of each model under the same prompting format. Self-Discover (Zhou et al., 2024) represents structure-driven prompting, while Self-Consistency (Wang et al., 2023; Chen et al., 2024) tests whether repeated sampling and voting can recover similar gains by spending more test-time computation. We also compare with DEER (Yang et al., 2026), an efficient dynamic-reasoning baseline, and SEAL (Chen et al., 2025b), a representative activation-steering baseline. Together, these baselines cover prompt-level structure, sampling-based compute scaling, efficiency-oriented reasoning control, and representation-level steering. Appendix B.2 reports a beam-style diagnostic that uses the same short-horizon confidence metric as STIR but branches over token continuations instead of applying latent corrections.

Implementation Details. All implementations are built on EasySteer (Xu et al., 2025) with a vLLM backend (Kwon et al., 2023) for high-throughput inference. During offline construction, each mining example is sampled with $K = 8$ stochastic rollouts, scored by correctness and length, and aligned at structural checkpoints. The resulting positive and negative prefixes are used to build a compact quality-diversity tool library, while evaluation examples remain disjoint from the mining set. At inference time, STIR uses the same base prompt and decoding setup as the corresponding Vanilla run, adding only retrieval, short-horizon probing, and bounded latent intervention. We report accuracy and average generated tokens as the primary metrics; dataset split sizes, generation budgets, and hyperparameters appear in Appendix A.2.

5.2 Main Results

Table 1 reports the main accuracy-token comparison. STIR improves the average accuracy of the accuracy-oriented setting ($k_{scale} = 0.75$) over Vanilla in all four main model blocks, with gains ranging from +1.9 to +7.5 points. The gains

Table 1: Main results across six datasets and four main model settings. Values represent Accuracy% (Avg. Tokens). The final column summarizes average accuracy change and average token reduction relative to Vanilla. Bold denotes the highest accuracy per block. STIR is reported with two strength settings to expose the accuracy-token trade-off induced by intervention strength.

Model	Method	Math Reasoning				General QA		Average Stats
		AIME 24	AIME 25	AMC 23	MATH500	ARC-C	OBQA	Acc (Δ) / Cost (Drop%)
DeepSeek-R1 Distill-Qwen 1.5B	Vanilla	40.0 (11,282)	30.0 (11,642)	60.0 (7,103)	72.0 (4,688)	59.2 (652)	50.0 (668)	51.9 (–) / 6,006 (1.0x)
	Self-Consistency	35.0 (32,761)	35.0 (28,535)	73.3 (17,529)	84.0 (9,441)	54.5 (2,065)	53.8 (2,089)	55.9 (+4.1) / 15,403 (x2.6)
	Self-Discover	20.0 (10,837)	20.0 (9,884)	50.0 (7,955)	74.0 (3,979)	57.2 (650)	47.2 (662)	44.7 (–7.1) / 5,661 (↓5%)
	DEER	30.0 (9,935)	40.0 (9,208)	66.7 (6,509)	71.0 (2,364)	56.9 (625)	51.2 (656)	52.6 (+0.8) / 4,883 (↓18%)
	SEAL	30.0 (11,802)	20.0 (13,262)	60.0 (7,495)	62.0 (5,458)	49.0 (665)	50.0 (640)	45.2 (–6.7) / 6,554 (↑9%)
	STIR $_{k_{scale}=1.0}$	25.0 (11,265)	30.0 (11,031)	70.0 (6,835)	71.5 (3,282)	49.8 (444)	47.8 (446)	49.0 (–2.8) / 5,550 (↓7%)
	STIR $_{k_{scale}=0.75}$	35.0 (11,930)	35.0 (11,383)	70.0 (6,645)	74.8 (4,297)	56.2 (669)	51.4 (629)	53.7 (+1.9) / 5,926 (↓1%)
DeepSeek-R1 Distill-Qwen 7B	Vanilla	55.0 (11,240)	45.0 (9,622)	86.7 (5,009)	85.2 (2,760)	77.9 (604)	75.4 (580)	70.9 (–) / 4,969 (1.0x)
	Self-Consistency	65.0 (27,942)	45.0 (29,249)	90.0 (14,587)	89.8 (7,974)	79.9 (1,889)	77.4 (1,905)	74.5 (+3.6) / 13,924 (x2.8)
	Self-Discover	40.0 (9,548)	45.0 (9,971)	80.0 (3,788)	84.5 (2,133)	76.3 (520)	73.8 (527)	66.6 (–4.3) / 4,415 (↓11%)
	DEER	70.0 (9,560)	50.0 (9,070)	83.3 (4,327)	80.8 (2,069)	77.3 (572)	74.0 (550)	72.6 (+1.7) / 4,358 (↓12%)
	SEAL	60.0 (8,902)	50.0 (9,417)	90.0 (3,728)	85.0 (3,189)	77.0 (679)	74.0 (643)	72.7 (+1.8) / 4,426 (↓10%)
	STIR $_{k_{scale}=1.0}$	65.0 (9,151)	50.0 (9,288)	90.0 (4,148)	87.0 (2,732)	79.9 (534)	73.2 (374)	74.2 (+3.3) / 4,371 (↓12%)
	STIR $_{k_{scale}=0.75}$	65.0 (8,941)	60.0 (9,653)	93.3 (4,000)	88.0 (2,638)	82.9 (600)	76.8 (591)	77.7 (+6.8) / 4,404 (↓11%)
Qwen2.5 3B-Instruct	Vanilla	5.0 (2,749)	5.0 (2,559)	43.3 (1,416)	65.8 (872)	80.9 (297)	75.4 (263)	45.9 (–) / 1,359 (1.0x)
	Self-Consistency	10.0 (2,991)	0.0 (3,090)	36.7 (2,796)	63.2 (1,918)	83.9 (906)	77.6 (804)	45.2 (–0.7) / 2,084 (↑53%)
	Self-Discover	10.0 (2,497)	5.0 (1,021)	56.7 (1,446)	62.7 (873)	73.9 (323)	70.0 (289)	46.4 (+0.5) / 1,075 (↓20%)
	DEER	10.0 (2,084)	0.0 (2,186)	50.0 (1,566)	61.0 (1,146)	75.9 (354)	69.2 (329)	44.4 (–1.5) / 1,278 (↓6%)
	SEAL	0.0 (2,802)	0.0 (2,571)	50.0 (853)	70.0 (1,002)	85.0 (313)	66.0 (276)	45.2 (–0.7) / 1,303 (↓4%)
	STIR $_{k_{scale}=1.0}$	10.0 (1,509)	0.0 (981)	43.3 (875)	65.8 (752)	75.9 (265)	76.4 (256)	45.2 (–0.7) / 773 (↓43%)
	STIR $_{k_{scale}=0.75}$	20.0 (1,264)	15.0 (1,023)	56.7 (1,432)	67.5 (984)	84.3 (287)	77.2 (259)	53.4 (+7.5) / 875 (↓35%)
Qwen2.5 7B-Instruct	Vanilla	20.0 (1,305)	10.0 (1,045)	53.3 (1,853)	74.5 (731)	88.0 (304)	78.2 (279)	54.0 (–) / 920 (1.0x)
	Self-Consistency	20.0 (2,998)	25.0 (2,983)	43.3 (2,828)	75.5 (1,919)	88.6 (925)	79.4 (844)	55.3 (+1.3) / 2,083 (x2.3)
	Self-Discover	15.0 (1,865)	15.0 (878)	60.0 (821)	71.0 (709)	79.6 (326)	73.6 (280)	52.4 (–1.6) / 813 (↓11%)
	DEER	20.0 (1,315)	15.0 (986)	50.0 (935)	73.0 (822)	88.3 (321)	77.8 (294)	54.0 (+0.0) / 779 (↓15%)
	SEAL	20.0 (992)	10.0 (1,078)	63.3 (672)	80.0 (717)	86.0 (320)	79.0 (279)	56.4 (+2.4) / 676 (↓26%)
	STIR $_{k_{scale}=1.0}$	20.0 (965)	5.0 (911)	56.7 (797)	70.0 (622)	88.0 (299)	76.8 (239)	52.7 (–1.3) / 639 (↓30%)
	STIR $_{k_{scale}=0.75}$	25.0 (1,096)	20.0 (937)	66.7 (888)	75.5 (778)	89.0 (305)	80.4 (254)	59.4 (+5.4) / 710 (↓22%)

are especially clear on the larger DeepSeek and Qwen settings: DeepSeek-R1-Distill-Qwen-7B improves from 70.9% to 77.7% average accuracy while reducing average tokens from 4,969 to 4,404, and Qwen2.5-7B-Instruct improves from 54.0% to 59.4% while reducing average tokens by 22%. On Qwen2.5-3B-Instruct, where Vanilla is weaker on competition math, STIR obtains the largest average gain (+7.5 points) while reducing token cost by 35%. The token column is therefore part of the main comparison, not merely an efficiency add-on, because STIR targets single-trajectory improvement under a bounded generation budget.

The comparison highlights a different trade-off from sampling-based test-time scaling. Self-Consistency remains a strong way to improve reliability through repeated sampling, but its gains require substantially longer generation. On DeepSeek-R1-Distill-Qwen-7B, Self-Consistency reaches 74.5% average accuracy with 13,924 average tokens, whereas STIR reaches 77.7% with 4,404 tokens. The efficiency-oriented STIR setting ($k_{scale} = 1.0$) often reduces token cost more aggressively, although it can sacrifice accuracy on harder model-task combinations. These patterns

motivate reporting accuracy and token cost jointly rather than optimizing either metric alone.

5.3 Geometry of Contrastive Latent States

We next inspect whether successful and failed rollout states provide useful signals for local corrections. Figure 2 projects latent states from MATH500 and OBQA rollouts into two dimensions using t-SNE (Maaten and Hinton, 2008). The projected states overlap globally, but many local neighborhoods still contain consistent directions from failed prefixes toward successful prefixes. The visualization therefore supports a neighborhood-level interpretation rather than a global separability argument.

This local organization is compatible with STIR’s controller because the method does not require one universal steering direction. The library stores localized contrasts keyed by hidden state and checkpoint index, so overlapping global geometry remains compatible with effective local corrections. The controller only needs a nearby displacement candidate for the current state, not a dataset-level classifier that separates all successes from all failures. This evidence supports mining paired rollout

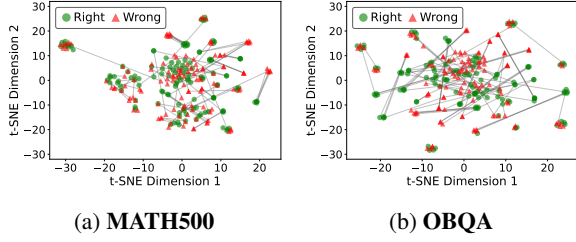


Figure 2: t-SNE visualization of contrastive latent states. Red triangles denote failure states, green circles denote successful states, and directed edges indicate mined correction vectors between local contrasts.

contrasts and routing them state by state instead of applying one static vector to the full trajectory.

Figure 3 reports PCA diagnostics for contrastive latent states. OBQA shows clearer right/wrong organization in the leading components, while MATH500 has more overlap. These plots motivate local contrastive routing: even when global separation is weak, nearby success-failure contrasts can still provide useful correction directions. Thus, the PCA view supports local routing rather than a global classifier over outcomes.

5.4 Sensitivity Analysis

We also examine the sensitivity of STIR to two control parameters: injection strength k_{scale} and intervention layer depth. Figure 4 shows that AMC23 is more sensitive to both choices, while MATH500 varies more mildly. Across these settings, very weak perturbations often fail to move the model away from its default trajectory, while overly strong perturbations can over-correct and reduce accuracy. The useful region is therefore a moderate intervention range rather than the largest possible steering magnitude.

Layer placement shows a similar calibration effect. Intervening too early risks editing low-level contextual features before the reasoning state has stabilized, while intervening too late leaves little opportunity for the correction to affect subsequent reasoning. The best region in Figure 4 lies around intermediate normalized depths, consistent with the view that multi-step reasoning decisions are most steerable after the model has formed a task representation but before final token prediction dominates. We use these sweeps to choose a stable operating range for the main experiments, while allowing the final setting to vary across models and datasets.

Table 2: Ablation study on DeepSeek-R1-Distill-Qwen-7B. We report accuracy (%) and average tokens after removing individual offline and online components.

Configuration	AMC 23		MATH 500	
	Acc.	Avg. Cost	Acc.	Avg. Cost
STIR (Full)	93.3	4000	88.0	2639
<i>Offline Library Construction</i>				
w/o Contrastive Tool Induction	83.3 (-10.0)	5041 (+1041)	83.0 (-5.0)	2904 (+265)
w/o Diversity (Top-k)	76.7 (-16.6)	6048 (+2048)	87.3 (-0.7)	2543 (-96)
<i>Online Control Policy</i>				
w/o Lookahead Probing	90.0 (-3.3)	5180 (+1180)	81.0 (-7.0)	2096 (-543)
w/o Anchor Gating	86.7 (-6.6)	5203 (+1203)	85.8 (-2.2)	2855 (+216)
w/o Adaptive Injection	93.3 (-0.0)	4488 (+488)	80.3 (-7.7)	3405 (+766)

5.5 Ablation Study

To validate the architectural decisions in STIR, we conduct a component ablation on DeepSeek-R1-Distill-Qwen-7B using AMC23 and MATH500. Table 2 shows that both library content and routing policy contribute materially to performance. Removing Contrastive Tool Induction reduces accuracy by 10.0 points on AMC23 and 5.0 points on MATH500, indicating that the correction vectors cannot be replaced by arbitrary perturbations. Removing diversity selection also degrades performance, especially on AMC23, suggesting that redundant tools fail to cover distinct failure modes.

The online policy is equally important. Removing lookahead probing reduces accuracy, showing that nearest-neighbor retrieval alone can select plausible but unhelpful corrections. Removing anchor gating also hurts performance and increases cost on both datasets, matching the intended role of anchors: when the current state already resembles a successful prefix, the appropriate action is often to abstain. Finally, removing adaptive injection leaves the controller unable to scale intervention magnitude to local evidence, producing a particularly large drop on MATH500. These results support the full Lookahead-Gated Intervention policy over a simpler always-steer policy.

5.6 Cross-Task Reuse of Latent Tools

To investigate whether distilled latent tools capture reusable reasoning patterns, we evaluate libraries built from one source dataset on other target datasets. Figure 5 shows the resulting reuse matrix. The strongest off-diagonal behavior appears within related mathematical tasks, where tools mined from AIME, AMC23, or MATH500 can still provide useful corrections on other math benchmarks. We also observe several cross-domain cases with im-

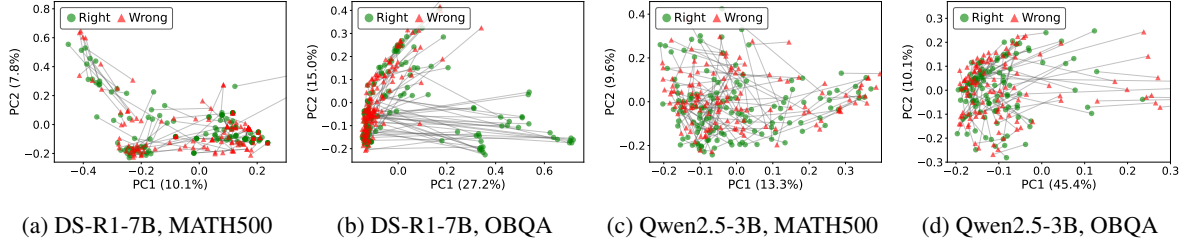


Figure 3: PCA visualization of contrastive latent states. The plots provide diagnostic evidence of local right/wrong organization, with clearer separation on OBQA and more overlap on MATH500.

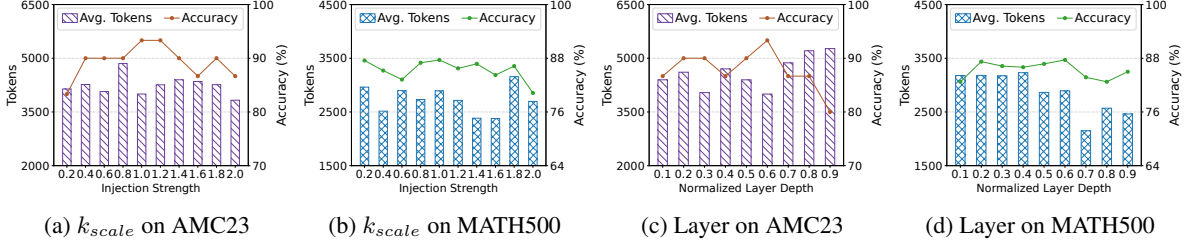


Figure 4: Sensitivity analysis of key control hyperparameters. Subplots (a) and (b) vary injection strength k_{scale} ; subplots (c) and (d) vary normalized layer depth. AMC23 shows the clearest dependence on strength and layer, while MATH500 is comparatively smoother.

Table 3: Computational overhead analysis across model scales. Avg. tokens denotes the mean cumulative count of additional tokens required by Lookahead-Gated Intervention, while ratio reports this overhead as a percentage of the total inference cost.

Dataset	DeepSeek-R1-1.5B		DeepSeek-R1-7B	
	Avg. (Total) Tokens	Ratio	Avg. (Total) Tokens	Ratio
AIME24	138.80 (11930)	1.16%	130.51 (8941)	1.46%
AIME25	134.80 (11383)	1.18%	138.54 (9653)	1.44%
AMC23	140.00 (6645)	2.11%	56.00 (4000)	1.40%
MATH500	128.79 (4297)	3.00%	132.84 (2638)	5.03%

proved accuracy or reduced token cost, suggesting that some tools encode general operations such as checking constraints or decomposing a problem.

The reuse matrix also shows clear source dependence. Some source-target pairs mainly change token cost or yield only small accuracy shifts, indicating that latent tools remain partly tied to the distribution from which they are mined. This is expected if tools encode local operations: reuse appears when a target needs the same operation and weakens under source-specific shortcuts. Appendix B.4 reports selected source-target pairs relative to Vanilla for easier numerical comparison.

5.7 Computational Overhead Analysis

We quantify the online cost of STIR by tracking the additional probe tokens introduced by Lookahead-Gated Intervention. Table 3 shows that this over-

head is small relative to the full reasoning trace: across the evaluated DeepSeek settings, the added probe budget is typically around 56 to 140 tokens, corresponding to roughly 1.1% to 5.0% of the total generated-token cost.

This overhead remains limited because lookahead probes are short and can be batched over candidate tools with shared prefix states. STIR adds validation before steering, but its added cost scales with probe horizon rather than full output length, making the overhead local rather than sample-wide. A fuller wall-clock decomposition, including the reusable offline library build cost, is provided in Appendix B.5.

5.8 Reasoning Length Distribution

We analyze AMC23 output lengths as an efficiency diagnostic. Figure 6 shows that STIR often shifts outputs away from very long generations, especially on the DeepSeek-R1-Distill-Qwen settings. This pattern is consistent with length-regularized mining and checkpoint-level abstention, which preserve useful reasoning steps while limiting repeated continuation. Because average cost can be dominated by a small number of long chains, the histograms clarify whether savings come from broad shortening or from suppressing the long tail of repeated verification loops.

Figure 6 complements the average token counts in Tables 1 and 3: averages cannot distinguish

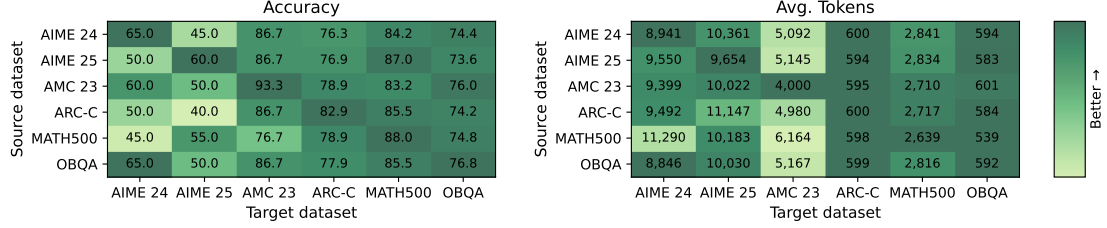


Figure 5: Cross-task reuse analysis. Rows indicate the source dataset used to construct the library, and columns indicate the target evaluation dataset. The left panel reports accuracy, while the right panel reports average generated tokens. The off-diagonal entries show that some latent tools transfer across related tasks, with effectiveness depending on source-target relatedness.

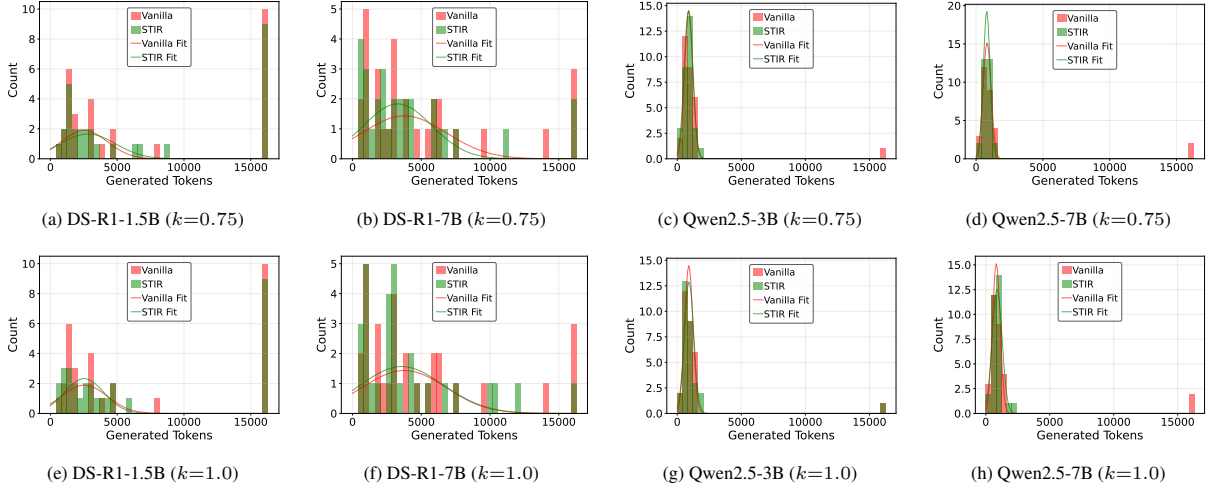


Figure 6: Token usage distribution on AMC23. We compare STIR (green) and Vanilla (red) across four model settings with $k_{scale} = 0.75$ (top) and $k_{scale} = 1.0$ (bottom). The plots show that STIR often reduces the long-token tail, though the magnitude varies by model and strength setting.

broad shifts from a few extreme generations. The central mass and the long-token tail change by different amounts across backbones, and the two intervention-strength settings do not induce a uniform shift. We therefore interpret efficiency together with accuracy rather than treating shorter output as an objective by itself. A useful intervention may preserve a longer chain when the task still requires verification, whereas anchor-based abstention avoids spending probe and generation budget when the current trajectory already resembles a successful state. Taken together with the overhead measurements, these distributions indicate that STIR’s cost profile is shaped by two local effects: a bounded probing expense at selected checkpoints and model-dependent changes in continuation length. This joint view distinguishes typical outputs from rare extremes. The overlap in the central bins further shows that STIR does not uniformly shorten every response. Visible differences are concentrated mainly in longer continuations.

6 Conclusion

We introduced STIR, a test-time framework that casts activation steering as a routed library of latent reasoning tools. STIR mines contrastive correction vectors from the model’s own rollouts, sparsifies them into a diverse library, and applies a candidate only after state retrieval and short-horizon probing. Across six benchmarks, STIR improves accuracy-token trade-offs over Vanilla, with gains depending on local geometry, component design, calibration, and conditional reuse rather than repeated full generations. By coupling sparse retrieval, abstention, and lookahead validation, STIR reframes steering as selective control over intermediate computation rather than as a global edit. Latent reasoning control should be local, evidence-conditioned, and cost-aware: interventions are most appropriate when a retrieved contrast matches the current checkpoint and probe evidence predicts a better continuation; otherwise, the controller should preserve the model’s intrinsic trajectory.

Limitations

STIR has two main limitations. First, the method depends on an offline library built from labeled correctness and stochastic rollouts. This cost can be amortized when a library is reused, but transfer remains source-dependent: some source-target pairs improve accuracy or token cost, whereas others show only small or mixed effects. Second, STIR is a local intervention method rather than a guarantee of trajectory correctness. Its checkpoint heuristic may miss errors inside dense reasoning segments, and activation edits can still perturb otherwise correct continuations. Anchor gating and lookahead probing reduce this risk, but broader backbone coverage and more robust checkpointing remain important future work.

Acknowledgments

Congcong Miao is supported by the Singapore Ministry of Education, under Academic Research Fund Tier 3 grant MOE-MOET32024-0004.

References

- Seyedarmin Azizi, Erfan Baghaei Potraghloo, Souvik Kundu, and Massoud Pedram. 2026. Activation steering for chain-of-thought compression. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 36676–36687, San Diego, California, United States. Association for Computational Linguistics.
- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.
- Lingjiao Chen, Jared Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. 2024. Are more llm calls all you need? towards the scaling properties of compound ai systems. *Advances in Neural Information Processing Systems*, 37:45767–45790.
- Runjin Chen, Andy Ardit, Henry Sleight, Owain Evans, and Jack Lindsey. 2025a. Persona vectors: Monitoring and controlling character traits in language models. *arXiv preprint arXiv:2507.21509*.
- Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. 2025b. SEAL: Steerable reasoning calibration of large language models for free. In *The Second Conference on Language Modeling*.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.
- Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. 2012. Near-optimal map inference for determinantal point processes. *Advances in Neural Information Processing Systems*, 25.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. 2025. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638.
- Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2025. Training large language models to reason in a continuous latent space. In *The Second Conference on Language Modeling*.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.
- Yao Huang, Huanran Chen, Shouwei Ruan, Yichi Zhang, Xingxing Wei, and Yinpeng Dong. 2025. Mitigating overthinking in large reasoning models via manifold steering. *Advances in Neural Information Processing Systems*, 38:113645–113670.
- Eunki Kim, Sangryul Kim, and James Thorne. 2025. Learning to insert [pause] tokens for better reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 23760–23777, Vienna, Austria. Association for Computational Linguistics.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626. Association for Computing Machinery.
- Tianhe Lin, Jian Xie, Siyu Yuan, and Deqing Yang. 2025. Implicit reasoning in transformers is reasoning through shortcuts. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 9470–9487, Vienna, Austria. Association for Computational Linguistics.
- Charles London and Varun Kanade. 2025. Pause tokens strictly increase the expressivity of constant-depth transformers. *Advances in Neural Information Processing Systems*, 38:99681–99711.

- Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605.
- math-ai. 2023. American mathematics competition 2023 dataset.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.
- Kiho Park, Yo Joong Choe, and Victor Veitch. 2024. The linear representation hypothesis and the geometry of large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 39643–39666. PMLR.
- Nina Rimskey, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. 2024. Steering Llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand. Association for Computational Linguistics.
- Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. 2025. CODI: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 677–693, Suzhou, China. Association for Computational Linguistics.
- Zhenning Shi, Dan Zhao, Yijia Zhu, Guorui Xie, Qing Li, and Yong Jiang. 2025a. Helios: Learning and adaptation of matching rules for continual in-network malicious traffic detection. In *Proceedings of the ACM on Web Conference 2025*, pages 2319–2329. Association for Computing Machinery.
- Zhenning Shi, Yijia Zhu, Yi Xie, Junhan Shi, Guorui Xie, Haotian Zhang, Yong Jiang, Congcong Miao, and Qing Li. 2025b. Reasoning under uncertainty: Efficient LLM inference via unsupervised confidence dilution and convergent adaptive sampling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32204–32218, Suzhou, China. Association for Computational Linguistics.
- Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. 2025. Layer by layer: Uncovering hidden representations in language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 55854–55875. PMLR.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*.
- Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. 2025. Improving instruction-following in language models through activation steering. In *The Thirteenth International Conference on Learning Representations*.
- Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, et al. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*.
- Eric Todd, Millicent L Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. 2024. Function vectors in large language models. In *The Twelfth International Conference on Learning Representations*.
- Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. 2023. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.
- Jingyu Xiao, Jiantong Qin, Shuoqi Li, Man Ho Lam, Yuxuan Wan, Jen-tse Huang, Yintong Huo, and Michael R. Lyu. 2026a. Component-based reusable UI code generation for complex websites via semantic segmentation and fine-grained feedback. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 5638–5649. Association for Computing Machinery.
- Jingyu Xiao, Yuxuan Wan, Yintong Huo, Zixin Wang, Xinyi Xu, Wenxuan Wang, Zhiyao Xu, Yuhang Wang, and Michael R. Lyu. 2025a. Interaction2Code: Benchmarking MLLM-based interactive webpage code generation from interactive prototyping. In *2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 241–253. IEEE Press.
- Jingyu Xiao, Ming Wang, Man Ho Lam, Yuxuan Wan, Junliang Liu, Yintong Huo, and Michael R. Lyu. 2025b. DesignBench: A comprehensive benchmark for MLLM-based front-end code generation. *arXiv preprint arXiv:2506.06251*.
- Jingyu Xiao, Zhongyi Zhang, Haoran Hou, Yuxuan Wan, Yuan Jiang, Yintong Huo, and Michael R. Lyu.

- 2026b. VisualRepair: Dynamic tool calling and region focusing for visual software issue repair. *arXiv preprint arXiv:2607.14075*.
- Jingyu Xiao, Zhongyi Zhang, Yuxuan Wan, Yintong Huo, Yang Liu, and Michael R. Lyu. 2026c. EfficientUICoder: A bidirectional token compression framework for efficient MLLM-based UI code generation. *Proceedings of the ACM on Software Engineering*, 3(FSE):2396–2418.
- Haolei Xu, Xinyu Mei, Yuchen Yan, Rui Zhou, Wenqi Zhang, Weiming Lu, Yueting Zhuang, and Yongliang Shen. 2025. Easysteer: A unified framework for high-performance and extensible llm steering. *arXiv preprint arXiv:2509.25175*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. 2026. Dynamic early exit in reasoning models. In *The Fourteenth International Conference on Learning Representations*.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.
- Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. 2024. Quiet-STaR: Language models can teach themselves to think before speaking. In *The First Conference on Language Modeling*.
- Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488.
- Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Yunhua Zhou, and Xipeng Qiu. 2025. Revisiting the test-time scaling of o1-like models: Do they truly possess test-time scaling capabilities? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4651–4665, Vienna, Austria. Association for Computational Linguistics.
- Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, et al. 2025. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.
- Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V Le, Ed Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. 2024. Self-discover: Large language models self-compose reasoning structures. *Advances in Neural Information Processing Systems*, 37:126032–126058.
- Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, et al. 2025. A survey on latent reasoning. *arXiv preprint arXiv:2507.06203*.

A Implementation Details

This appendix provides the technical specifications needed to reproduce STIR. It summarizes notation, hyperparameters, and algorithmic details for Contrastive Tool Induction, Sparse Tool Library Construction, and Lookahead-Gated Intervention, complementing the main methodology in Section 4.

A.1 Notation Summary

Table 4 summarizes the mathematical symbols used to describe the interaction between the language model and the tool library. The notation covers residual-stream states, sparse-library geometry, retrieval scores, probing gains, and gating thresholds. It also distinguishes correction entries, which carry active steering impulses, from anchor entries, which represent high-quality local states where the controller should abstain.

A.2 Hyperparameters

Table 5 lists the hyperparameters used in STIR. For Contrastive Tool Induction, we use $K = 8$ stochastic rollouts per mining example and a length penalty coefficient $\eta = 0.01$. Sparse Tool Library Construction uses a maximum library size $B = 256$ and diversity weight $\lambda = 0.5$. For Lookahead-Gated Intervention, the retrieval top- k is 8 and the lookahead horizon is $T_{probe} = 4$, so validation remains lightweight while still providing an instance-specific signal.

A.3 Algorithms and Engineering

STIR is organized into offline library construction and online Lookahead-Gated Intervention. Algorithms 1 and 2 make this division explicit and instantiate the three components described in Section 4.

Offline Library Construction. Algorithm 1 distills a compact set of steering primitives from stochastic model rollouts. Each rollout is evaluated

Table 4: Mathematical notations and their descriptions used in the STIR framework.

Symbol	Description
$\mathcal{M}_\theta$	Target transformer-based Large Language Model.
$h_t^{(l)}$	Latent hidden state at timestep t and layer l .
$\pi = (k, v, q)$	Steering tool unit comprising a state key, steering impulse, and quality score.
$k \in \mathbb{R}^d$	State key (context anchor) used for tool retrieval.
$v \in \mathbb{R}^d$	Steering impulse (directional vector) for trajectory correction.
$q(\pi)$	Historical quality score derived from reward gain during offline discovery.
$\mathcal{M}$	Sparse tool library used for retrieval during intervention.
Ω	Raw candidate set of discovered tools before sparse selection.
$\mathcal{C}_t$	Locally retrieved candidate set of steering tools at timestep t .
$\hat{A}(\pi)$	Similarity-weighted quality score used as the retrieval prior.
$\mathcal{G}(\pi)$	Lookahead gain estimated by short counterfactual probing.
$S(\pi)$	Unified utility score for final tool selection and gating.
τ_{null}	Hard abstention threshold (null action threshold).
k_{scale}	Scaling factor for adaptive injection strength.
β, ρ	Weighting coefficients for prior score and probing gain, respectively.
λ	Diversity weight in the Sparse Tool Library Construction objective.
T_{probe}	Horizon length (tokens) for lookahead previewing.
B	Total library budget (maximum entries in the library).

Table 5: Hyperparameter configurations employed in the STIR framework.

Phase	Hyperparameter	Value
Contrastive Tool Induction	Rollout count per example (K)	8
	Positive pool size (k_{pos})	2
	Negative pool size (k_{neg})	3
	Length penalty coefficient (η)	0.01
	Sampling temperature	0.7
Sparse Tool Library Construction	Total library size (B)	256
	Diversity weight (λ)	0.5
	DPP stability term (ϵ)	10^{-4}
Lookahead-Gated Intervention	Retrieval top- k	8
	Candidate pool size (L)	4
	Lookahead horizon (T_{probe})	4
	Prior weight (β)	2.0
	Probing weight (ρ)	0.1
	Normalized layer depth	0.6
	Strength scale (k_{scale})	1.0

with the length-regularized reward in Eq. 3. At structural checkpoints, trajectories are partitioned into positive and negative prefix sets, converted into latent centroids using Eq. 4, and contrasted to obtain candidate steering impulses. Sparse Tool Library Construction then uses marginal-gain determinantal selection (Gillenwater et al., 2012), favoring high-utility entries while reducing redundancy under Eq. 6. All state keys are L2-normalized, which keeps online retrieval based on cosine similarity consistent across layers.

Online Lookahead-Gated Intervention. Algorithm 2 implements Lookahead-Gated Intervention. At each decision point, the controller retrieves candidate tools by angular similarity to the current

hidden state and first checks whether anchor evidence supports abstention. For the remaining active candidates, a short-horizon counterfactual probe estimates the dynamic reward gain in Eq. 7. The controller combines retrieval and probe evidence into a unified utility score, applies a null-threshold check, and injects the selected steering impulse only when the local evidence is strong enough. The perturbation magnitude is then scaled by decision confidence according to Eq. 8, preventing every retrieved tool from becoming a fixed-strength edit.

Engineering Infrastructure. The implementation uses EasySteer (Xu et al., 2025) with a vLLM backend (Kwon et al., 2023). Offline rollout generation can be parallelized across GPUs. During on-

Table 6: Mining/evaluation split sizes and maximum generation budgets.

Dataset	Mining	Eval	$T_{\max}$
AIME25	10	20	16384
AIME24	10	20	16384
AMC23	10	30	16384
ARC-C	100	299	1024
MATH500	100	400	16384
OBQA	100	500	1024

line inference, lookahead probes are batched over candidate tools and reuse the current generation context when possible. This keeps validation overhead closer to short local probing than to running separate full generations for every candidate.

B Additional Results and Diagnostics

B.1 Mining and Evaluation Splits

Table 6 reports the mining and evaluation sizes used to construct and test the STIR libraries. The protocol keeps mining and evaluation examples disjoint: mining examples produce stochastic rollouts for library construction, while evaluation examples are reserved for final testing and are never used to build the library. For datasets whose public release uses the same split name for both subsets, the split name identifies the public source partition; the item-level mining and evaluation subsets are sampled separately.

B.2 Search-Only Diagnostic Baseline

Because STIR uses a short lookahead probe, we include a search-only diagnostic that uses the same confidence signal (Shi et al., 2025b) but branches over token continuations instead of applying latent corrections. Table 7 shows that lookahead confidence alone does not explain STIR’s gains on DeepSeek-R1-Distill-Qwen-7B: Beam Search improves only modestly over Vanilla, while STIR reaches higher average accuracy at a comparable average token cost.

This comparison is intended as a diagnostic rather than as a full decoding study. It controls for the possibility that the probe signal simply behaves like a lightweight search heuristic. The gap between Beam Search and STIR suggests that the main benefit comes from applying a retrieved latent correction to the current reasoning state, not merely from spending the same probe budget on nearby token continuations.

B.3 Additional Model-Family Evidence

We view this run as additional model-family evidence. Under the same mining and intervention protocol, it evaluates whether the rollout-derived controller remains effective when the target backbone differs from the main Qwen-derived settings. The result therefore tests protocol-level transfer rather than only within-family reuse.

B.4 Cross-Dataset Reuse

Table 9 reports selected source-target reuse pairs relative to the Vanilla target model, complementing the full cross-dataset matrix in Figure 5. The selected rows give representative cases where reuse improves accuracy, reduces tokens, or trades one objective for the other.

The selected rows highlight two recurring behaviors. First, mathematically related libraries can transfer useful local checks across datasets, as in AIME25 $\rightarrow$ MATH500. Second, some cross-domain reuse is beneficial when the target problem requires a general reasoning operation such as constraint checking. The token column is included because reuse can improve accuracy while increasing cost, so both dimensions are needed to interpret transfer behavior.

B.5 Engineering Cost Decomposition

Table 10 separates online inference from offline library construction. The Self-Consistency column corresponds to three repeated Vanilla samples under the same decoding setup. STIR incurs an offline cost, but the resulting library can be reused across evaluation examples and related tasks. The decomposition separates per-query online overhead from amortizable preparation cost, which are operationally different.

The wall-clock results also clarify the difference between STIR and sampling-heavy alternatives. Self-Consistency scales almost linearly with the number of repeated samples, whereas STIR pays for short probes and latent-vector operations during a single controlled trajectory. The offline build time is therefore best viewed as a reusable indexing cost: it matters for a one-off deployment, but becomes less dominant when the same library is applied to multiple evaluation batches or reused for related analyses (Shi et al., 2025a).

B.6 Case Studies

We summarize representative interventions in prose rather than reproducing full generations. The cases

Algorithm 1 STIR Offline: Contrastive Tool Induction and Sparse Tool Library Construction

Require: Training set $\mathcal{D}$, model $\mathcal{M}_\theta$, checkpoints M , library budget B **Ensure:** Sparse tool library $\mathcal{M}$

```
1: // Stage I: Contrastive Tool Induction
2: for each example  $(x, y) \in \mathcal{D}$  do
3:   Sample  $K$  stochastic rollouts  $\{Y_1, \dots, Y_K\}$  from  $\mathcal{M}_\theta$ 
4:   Compute rewards  $\{R_1, \dots, R_K\}$  using Eq. (3)
5:   for each checkpoint  $m = 1$  to  $M$  do
6:     Identify positive prefixes  $\mathcal{P}_m$  and negative prefixes  $\mathcal{N}_m$ 
7:     Extract latent centroids  $\mu_m^+$  and  $\mu_m^-$  via Eq. (4)
8:      $v_m \leftarrow \mu_m^+ - \mu_m^-$  {Steering impulse}
9:     Calculate quality score  $q_m$  based on reward gain
10:    Store correction entry  $\pi = (\mu_m^-, v_m, q_m)$  and anchor entry  $(\mu_m^+, 0)$  in  $\Omega$ 
11:   end for
12: end for
13: // Stage II: Sparse Tool Library Construction
14: Select  $B$  entries from  $\Omega$  by maximizing Eq. (6)
15: // Stage III: Indexing
16: Normalize all keys  $k \in \mathcal{M}$  such that  $\|k\|_2 = 1$ 
17: return  $\mathcal{M}$ 
```

Table 7: Beam-style search diagnostic on DeepSeek-R1-Distill-Qwen-7B. Beam Search uses the same short-horizon confidence metric as STIR but does not apply activation steering. Avg. reports average accuracy / average tokens.

Method	AIME24	AIME25	AMC23	MATH500	ARC-C	OBQA	Avg. Acc / Tok
Vanilla	55.0	45.0	86.7	85.2	77.9	75.4	70.9 / 4969
Beam Search	60.0	50.0	86.7	76.5	82.6	74.8	71.8 / 4396
STIR	65.0	60.0	93.3	88.0	82.9	76.8	77.7 / 4404

illustrate three roles of the retrieved latent tools: recovering a missing algorithmic substep, encouraging invariance checking under cross-dataset reuse, and redirecting a shallow association toward a task-relevant constraint.

Recovering a missing arithmetic substep. On a MATH500 problem asking for the number of trailing zeros in $42!$, Vanilla follows the standard factorial-zero strategy but stops after counting multiples of 5, giving $\lfloor 42/5 \rfloor = 8$. STIR changes the local reasoning state so that the continuation also accounts for the additional factor of 5 contributed by 25, computing $\lfloor 42/5 \rfloor + \lfloor 42/25 \rfloor = 8 + 1 = 9$. This case illustrates the intended behavior of a local correction: the intervention keeps the overall solution strategy intact while restoring a missing step inside it.

Checking invariance in a math transfer case. In a cross-dataset reuse example from commonsense to mathematics, the target question involves distributing indistinguishable balls into boxes. Vanilla applies a stars-and-bars-like pattern for distinct boxes and returns six ordered allocations. After STIR intervention, the model treats box permutations as equivalent and enumerates the unordered partitions $(0, 5)$, $(1, 4)$, and $(2, 3)$, producing the

correct count of three. The qualitative change suggests that the retrieved tool encourages an invariance check, helping the model avoid a surface formula match when the problem objects are indistinguishable.

Redirecting a shallow association in commonsense transfer. In a math-to-commonsense transfer example, the prompt asks for the most appropriate completion of a desert-related question. Vanilla follows the salient lexical association between “desert” and “cactus.” With STIR, the continuation instead focuses on the decisive environmental constraint that water is scarce in deserts and selects “stream.” This example does not imply that a mathematical tool directly encodes commonsense knowledge; rather, it shows that cross-domain latent tools can sometimes act as local constraint-checking operators when the target problem requires rejecting an intuitive but under-specified association.

Table 8: Llama-3.1-8B-Instruct results. Accuracy is reported in percent and tokens are average generated tokens.

Method	AIME24	AIME25	AMC23	ARC-C	MATH500	OBQA	Avg. Acc / Tok
Vanilla Acc	0.0	5.0	30.0	81.3	52.3	79.8	41.4
STIR Acc	5.0	10.0	33.3	82.6	54.8	79.9	44.3
Vanilla Tok	6217	5369	6429	282	2012	256	3427
STIR Tok	3932	3890	4805	287	1970	250	2522

Table 9: Selected cross-dataset reuse results on DeepSeek-R1-Distill-Qwen-7B, reported relative to the Vanilla target model.

Source $\rightarrow$ Target	STIR Acc	Vanilla Acc	Δ Acc	STIR Tok	Vanilla Tok	Δ Tok
OBQA $\rightarrow$ AIME24	65.0	55.0	+10.0	8846	11240	-2394
AIME25 $\rightarrow$ MATH500	87.0	85.2	+1.8	2834	2760	+74
AMC23 $\rightarrow$ ARC-C	78.9	77.9	+1.0	595	604	-9
MATH500 $\rightarrow$ AIME25	55.0	45.0	+10.0	10183	9622	+561

Table 10: Wall-clock cost decomposition on DeepSeek-R1-Distill-Qwen-7B. Times are reported as h:mm:ss.

Dataset	Vanilla	Self-Consistency	SEAL	STIR Online	STIR Offline Build	STIR Total
MATH500	0:16:10	0:48:30	21:56:39	1:13:21	0:40:31	1:53:52
OBQA	0:01:59	0:05:57	3:36:05	0:52:27	0:27:04	1:19:31
ARC-C	0:01:26	0:04:18	1:43:48	0:38:39	0:18:03	0:56:42
AMC23	0:07:38	0:22:54	7:08:55	0:12:08	0:11:16	0:23:24
AIME25	0:08:26	0:25:18	7:12:19	0:12:21	0:15:37	0:27:58
AIME24	0:08:57	0:26:51	7:09:53	0:11:54	0:19:08	0:31:02

Algorithm 2 STIR Online: Lookahead-Gated Intervention

Require: Prompt z , current prefix p , checkpoint index m , tool library $\mathcal{M}$

Ensure: Controlled segment or abstention

- 1: Extract latent descriptor $\phi_{m,l}$ from the residual stream of p at layer l
 - 2: $\mathcal{C}_t \leftarrow$ Retrieve top- k entries from $\mathcal{M}$ conditioned on $\phi_{m,l}$
 - 3: $\hat{A}_{anchor} \leftarrow$ Extract maximum similarity score from retrieved anchor entries
 - 4: **if** $\mathcal{C}_t$ is empty **or** $\hat{A}_{anchor}$ is dominant **then**
 - 5: **return** abstain {Consistency-based gating to trust intrinsic generation}
 - 6: **end if**
 - 7: Filter $\mathcal{C}_t$ to retain top- L candidates based on prior $\hat{A}$
 - 8: **for** each candidate $\pi_i \in \mathcal{C}_t$ **do**
 - 9: Execute short-horizon preview for T_{probe} tokens using a shared KV-cache
 - 10: Estimate average log-prob gain $\mathcal{G}(\pi_i)$ via Eq. (7)
 - 11: $S(\pi_i) \leftarrow \beta \cdot \hat{A}(\pi_i) + \rho \cdot \mathcal{G}(\pi_i)$ {Compute unified utility score}
 - 12: **end for**
 - 13: $\pi^* \leftarrow \arg \max_{\pi_i \in \mathcal{C}_t} S(\pi_i)$
 - 14: **if** $S(\pi^*) < \tau_{null}$ **then**
 - 15: **return** abstain
 - 16: **else**
 - 17: $\alpha \leftarrow \min(k_{scale} \cdot S(\pi^*), \alpha_{max})$ {Adaptive strength with clipping}
 - 18: Apply steering impulse v_{π^*} to $h_t^{(l)}$ with strength α
 - 19: **return** Generate next segment via $\mathcal{M}_\theta(\cdot | h_t^{(l)} + \alpha v_{\pi^*})$
 - 20: **end if**
-