Demystifying Hidden-State Recurrence: Switchable Latent Reasoning with On-Policy Reinforcement Learning

Anonymous ACL submission

Abstract

Latent chain-of-thought compresses reasoning by replacing visible reasoning traces with continuous hidden-state recurrence, but existing formulations are difficult to optimize with standard on-policy reinforcement learning (RL) and hard to interpret causally. Our key insight is that a single pair of explicit boundary tokens can address both issues at once: discrete entry and exit anchors make the latent block compatible with standard on-policy RL, and the same anchors offer a natural foothold for mechanistic analysis. Motivated by this, we propose SWITCH, a switchable latent reasoning framework. The model emits `<swi>` to enter latent mode and `</swi>` to exit. Because the boundaries are ordinary discrete tokens, the GRPO policy ratio is well-defined at every decision point. The same anchors also expose the latent steps to direct probing and causal intervention. We train the model with a visible-to-latent curriculum and a Switch-GRPO objective that propagates gradients through recurrent latent computation. SWITCH consistently outperforms prior hidden-state-recurrence latent reasoning approaches at similar scale. Mechanistic analysis through the boundary tokens further reveals three findings: (i) `<swi>` is a sharply localised, learned switching policy rather than a stylistic artefact; (ii) the latent step it opens performs problem-specific, causally important computation rather than acting as an inert placeholder; and (iii) that computation is concentrated at a single hidden-state transition on entry. Together, these results show that hidden-state-recurrence latent reasoning is both RL-trainable and open to direct mechanistic analysis, including of *how* on-policy RL itself improves the model from the inside.

1 Introduction

Latent chain-of-thought (CoT) compresses the reasoning trace of a Large Language Model (LLM) by replacing visible text steps with continuous latent steps. A natural realization of this idea is *hidden-state recurrence*, introduced by Coconut (Hao et al., 2025) and adopted by subsequent work (Shen et al., 2025): the latent step keeps computation inside the LLM's own hidden space, feeding the previous step's last-layer hidden state back as the next input embedding. The latent computation thus runs in the LLM's existing representation space and reuses the same forward pass that produces the surrounding text, without introducing additional architectural components.

Two specific challenges have held the approach back. First, on-policy reinforcement learning, now a standard tool for aligning reasoning models with task rewards (DeepSeek-AI, 2025; OpenAI, 2024)—does not transfer cleanly to the latent setting: latent positions emit no tokens and so have no policy density, leaving methods such as GRPO undefined inside the latent block. Existing systems therefore either skip RL or run text-only training rollouts that diverge from the inference-time decoder. Second, the latent computation is hard to inspect: latent positions sit inside a continuous text continuation with no token that an analyst can grip, leaving it unclear whether the latent step performs task-relevant computation or merely acts as an inert filler that the surrounding text compensates for. We observe that both issues share a common root cause: the absence of an explicit boundary that marks where latent computation begins and ends.

This observation motivates our core idea: introduce a pair of explicit boundary tokens that demarcate the latent block. The model emits `<swi>` to enter latent mode and `</swi>` to exit, with hidden-state recurrence in between. The boundaries make latent reasoning a learned, per-step decision—the model chooses whether and when to invoke it—which is what *switchable* refers to in this paper. Two consequences follow: `<swi>` and `</swi>` are ordinary discrete tokens, so the GRPO ratio is well-defined at every text position (latent positions sim-

ply contribute no policy-gradient term); and the same boundaries serve as anchors for analysis, letting us read $p(\texttt{<swi>})$, probe the switch state from internal activations, and intervene on specific latent hidden states.

The second affordance lets us address a recurring concern about non-decoding "thinking" tokens (Pfau et al., 2024; Goyal et al., 2024): that latent positions might be non-functional placeholders the model has learned to bypass, with the surrounding text doing the actual work. Whether hidden-state-recurrence latents share this fate has remained an open question, and the boundary tokens are what let us answer it directly.

We package this idea as SWITCH (Fig. 1): the model is trained in three phases—an SFT stage that wraps high-entropy CoT spans in <swi>/</swi>, a curriculum that gradually replaces text inside <swi> blocks with <latent> positions (adapted from Hao et al., 2025), and a Switch-GRPO optimizer that propagates gradients through the recurrent latent computation (Sheng et al., 2024). Our contributions are as follows:

- SWITCH addresses both challenges with one primitive: learned <swi>/</swi> boundary tokens, paired with a Switch-GRPO optimizer, make on-policy RL well-defined and expose the latent computation to direct mechanistic analysis.
- On MATH-500, SWITCH reaches $\mathbf{79.3}\%$, $+\mathbf{25.7}$ points above the strongest Coconut-style baseline at the same scale; Switch-GRPO over the SFT-only checkpoint further halves the latent invocation rate while raising accuracy on invoked problems by $+\mathbf{12.6}$ points.
- Mechanistic analysis through the boundary tokens yields three converging takeaways about the switch policy and the latent step's computation (§5).

2 Related Work

Latent CoT can be split by what a latent token is. Hidden-state recurrence (Hao et al., 2025; Shen et al., 2025) feeds the previous step's last-layer hidden state back as the next input embedding; vocabulary mixtures (Zhang et al., 2025; Deng et al., 2025, 2026; Zheng et al., 2025) instead sample a top-k convex combination of vocabulary embeddings via Gumbel-Softmax. Mixtures are samplable and admit direct policy gradients, which has motivated recent RL work to abandon hidden-state recurrence; we instead show the recurrence can be RL-trained (§3.3) and verified mechanistically (§5). Related non-recurrent approaches include training-free switchable inference on a frozen reasoning LLM (Shi et al., 2026), adaptive test-time compute that always emits visible thinking (Snell et al., 2024; Chen et al., 2024), and non-decoding pause-style tokens (Goyal et al., 2024; Pfau et al., 2024; Deng et al., 2024; Tan et al., 2025). Detailed positioning of each line, together with the interpretability tools we apply in §5 including logit lens (nostalgebraist, 2020; Belrose et al., 2023), linear probing (Tenney et al., 2019; Belinkov, 2022), and causal activation interventions (Meng et al., 2022; Heimersheim and Nanda, 2024), is provided in Appendix J.

3 Method

Training a hidden-state-recurrence latent reasoner is hard because latent positions admit neither supervision targets nor a sampling distribution, leaving both cross-entropy SFT and standard policy-gradient RL undefined inside the block. SWITCH addresses this with a single primitive, the boundary tokens <swi>/</swi>, that gives every training stage a discrete handle on the otherwise-continuous latent block. Three phases share it: (i) SFT teaches the model when to emit <swi>/</swi>; (ii) a curriculum gradually replaces text inside the boundaries with <latent> positions while keeping the boundary signal intact; and (iii) Switch-GRPO uses the same boundaries to make the GRPO ratio well-defined at every text position, allowing on-policy RL through trajectories that contain latent steps. We refer to (i) + (ii) as *Switch-SFT* and (iii) as *Switch-GRPO*; full equations and algorithm boxes are in Appendix B and D.

3.1 Switchable Latent Reasoning

We extend the model's vocabulary with three special tokens: <swi> (enter latent), </swi> (exit latent), and <latent> (latent placeholder). At inference, the model decodes normally until it emits <swi>, runs at least $K_{\min}$ latent steps, and may then emit </swi> to exit. The minimum dwell $K_{\min}$ is needed because in Phase 2 every <latent> run terminates with </swi> at a fixed offset, and without forcing a few steps the trained model exits in one; we explain why mechanistically in §5.

Following Coconut (Hao et al., 2025), the input embedding inside a latent block is the previous

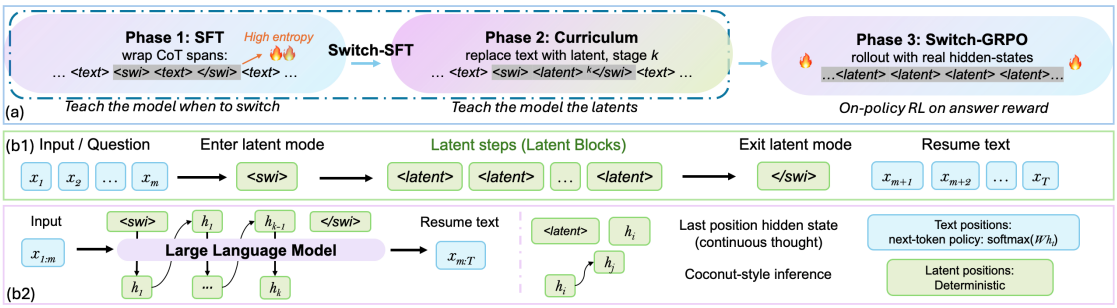

Figure 1: **SWITCH overview. (a) Training.** Three phases turn a Qwen3 base into a switchable latent reasoner: SFT to wrap high-entropy CoT spans in `<swi>`/`</swi>`, a curriculum that gradually replaces text inside those spans with `<latent>` positions (jointly, *Switch-SFT*), and Switch-GRPO for on-policy RL on the answer reward. **(b1) Inference token stream.** The model emits `<swi>` to enter latent mode, runs a block of `<latent>` steps, and emits `</swi>` to resume text decoding. **(b2) Hidden-state recurrence inside the block.** Each latent step's last-layer hidden state becomes the input embedding of the next `<latent>` position (Coconut-style recurrence).

step's last-layer hidden state:

$$\tilde{\boldsymbol{e}}_t = \begin{cases} E[x_t] & x_t \neq \texttt{<latent>}, \\ \boldsymbol{h}_{t-1} & x_t = \texttt{<latent>}. \end{cases} \tag{1}$$

Because $\boldsymbol{h}_{t-1}$ depends on $\tilde{\boldsymbol{e}}_{1:t-1}$, this is a recursion across latent positions: each latent step requires its own forward pass through the model, with the previous step's last-layer hidden state determining the next input embedding (implementation details in Appendix A). At text positions the next-token policy is the standard categorical $\mathrm{softmax}(W\boldsymbol{h}_t)$. At latent positions $\tilde{\boldsymbol{e}}_t$ is a Dirac mass and no token is sampled, which is why hidden-state-recurrence latents admit no direct policy density and what shapes the Switch-GRPO design below.

3.2 Switch-SFT: Curriculum Study

Phase 1 and Phase 2 together form what we will call the *Switch-SFT* stage: a two-step supervised fine-tuning recipe that takes a base model from visible CoT to a switchable latent reasoner. Phase 1 teaches the model when to enter and exit the latent block, and Phase 2 teaches it to do useful work inside that block. We report the resulting checkpoint as "after Switch-SFT" in §4.2 and use it as the initialisation for Phase 3.

Phase 1: locating switch positions. Phase 1 trains the model to mark *high-entropy* segments of a visible CoT with `<swi>`/`</swi>`. Following SwiReasoning (Shi et al., 2026), we measure high-entropy positions on a mathematical CoT corpus (Hugging Face, 2025b) as those where the base model's next-token distribution has high Shannon entropy—intuitively, positions where the model is uncertain about the next reasoning step—and tag contiguous runs of such positions with the boundary tokens. The annotated corpus is then used for standard next-token cross-entropy supervised fine-tuning over the response sequence (the prompt is masked from the loss as usual).

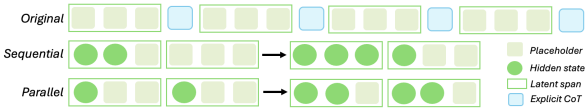

Figure 2: **Sequential vs. parallel curriculum schedules.** Hidden states (*green circles*) replace text inside `<swi>`-spans either one span at a time or in every span simultaneously across curriculum stages.

Phase 2: latent curriculum. Phase 2 progressively replaces text inside `<swi>`/`</swi>` blocks with `<latent>` positions while keeping `<swi>`/`</swi>` in the loss, so the model still has to learn when to enter and exit. A one-shot replacement is too aggressive: with no prior experience of latent computation, the model lets the block collapse into a no-op. We compared two schedules (Fig. 2). Let $S_1, \ldots, S_M$ be the `<swi>`-spans of a sample and $|S_m|$ the text length of span m. A *sequential* schedule converts spans one at a time, so at stage k only the leftmost k spans contain `<latent>` positions. A *parallel* schedule, our default, converts every span simultaneously and grows the per-span latent count:

$$n_m^{(k)} = c \cdot \min\big(k,\, |S_m|,\, K_{\max}\big), \tag{2}$$

with $c = 2$ and $K_{\max} = 8$. `<latent>` labels are masked, so the loss applies to non-latent positions.

The parallel schedule is substantially better in our runs. Our reading is that the sequential sched-

ule keeps most of each response inside the next-token-prediction distribution the base model was pre-trained on, with only one span deviating at a time, so the model can satisfy the loss without ever computing in latent space. The parallel schedule pushes every span out of that distribution at once and forces the model to produce hidden states the surrounding text has to condition on. The hyper-parameter sweep and the head-to-head comparison are in Appendix A.

3.3 Switch-GRPO: Latent Exploring

Phase 3 uses Group Relative Policy Optimization (GRPO) (Shao et al., 2024) to improve correctness and tag well-formedness. Two ingredients matter.

First, *Switch-GRPO redefines what a rollout is*. Standard GRPO assumes every rollout position is sampled from a categorical token distribution and contributes a policy density to the importance ratio. Hidden-state-recurrence latent execution violates this assumption: `<latent>` positions emit no token, no sampling distribution, and no density. Switch-GRPO resolves the conflict with two coupled changes. *(i) Rollout execution.* Rollouts run the same multi-pass forward as the deployed decoder, so the trajectories the optimiser sees at training time are exactly those produced at inference. Standard text-only RL pipelines (Sheng et al., 2024) silently bypass the latent step and train against a different inference path. *(ii) Likelihood factorisation.* Hidden-state injection is deterministic given the preceding text, so the rollout likelihood factors over text positions only. The GRPO ratio is therefore well-defined at every `<swi>`, `</swi>`, and visible answer token; latent positions contribute no policy-gradient term. Full equations are in Appendix B.

Second, *the reward is correctness-dominant but switch-aware*. We combine four terms in a weighted sum. A ± 1 correctness reward from `math-verify` (Hugging Face, 2025a) dominates the signal. A ± 1 tag-format reward enforces well-formed `<swi>`/`</swi>` pairs. A $\{0, 1\}$ latent-usage reward pays out when a correct answer uses `<swi>`, encouraging the model to invoke the latent path rather than fall back to plain text. The compression operating point of §4.2 adds an optional $[0, 1]$ correctness-gated brevity term. The full reward formula, the clipped surrogate loss, and the memory-segmented backward pass are in Appendix B.

4 Experiments

4.1 Experimental Setup

Model, Data and Benchmarks. All experiments, SWITCH and every baseline in Table 1 alike, use Qwen3-8B (Qwen Team, 2025) as the base model with three special tokens (`<swi>`, `</swi>`, `<latent>`) added to the vocabulary. Phases 1 and 2 use an annotated subset of OpenR1-Math (Hugging Face, 2025b) whose high-entropy CoT sub-spans are wrapped in `<swi>`/`</swi>` following the SwiReasoning annotation pipeline (Shi et al., 2026); Detailed training and hardware details are provided in Appendix A. For fair comparisons, we follow recent latent-CoT work (Deng et al., 2025, 2026) by using MATH-500 (Lightman et al., 2024; Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021) as the benchmarks.

Baselines. For Table 1, we re-implement every baseline on the same base model under matched data and decoding settings, following the protocols of CODI (Shen et al., 2025) and Latent-GRPO (Deng et al., 2026): a no-CoT direct-answer baseline, a text-CoT SFT baseline trained on the same corpus, two non-decoding "thinking" baselines (iCoT Deng et al., 2024, Pause Tokens Goyal et al., 2024), and three Coconut-style latent reasoning methods that share the same hidden-state-injection recurrence (Coconut Hao et al., 2025; CODI; CoLaR Tan et al., 2025). Re-implementation details are in Appendix A.

4.2 Main Results

This section is organized around the simplest version of the question: does Switch-GRPO actually learn anything over the Coconut curriculum alone, and if so, what does it learn? We first give the headline number against prior Coconut-style baselines, then compare the model immediately before and after RL to isolate what changed. A per-subject and per-difficulty breakdown is in Appendix G. Finally, we show that the reward also gives users an explicit accuracy–length knob.

Headline Performance Table 1 reports SWITCH and prior hidden-state-recurrence methods (§2), together with three standard non-latent references (no-CoT direct answer, text-CoT supervised fine-tuning, and the implicit/pause-token line). All baselines are evaluated on the same base model under matched data and decoding settings, so the comparison is apples-to-apples.

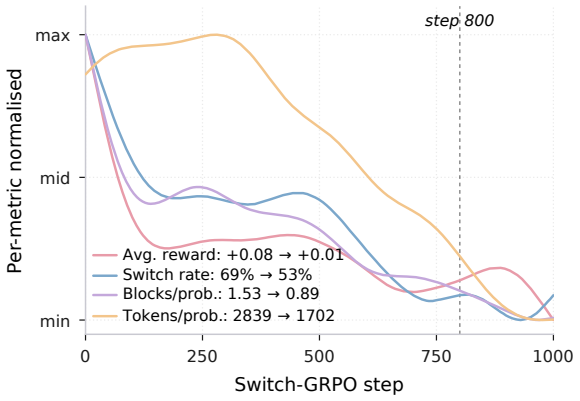

Figure 3: **Switch-GRPO training trajectory**, four metrics overlaid with per-metric min–max normalisation. Legend shows the actual start → end values; the dashed line marks the reported step-800 checkpoint.

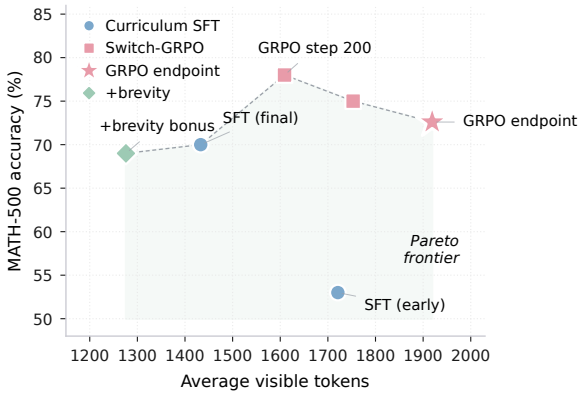

Figure 4: **Accuracy–efficiency operating curve** on the representative training run. MATH-500 accuracy vs. average visible tokens. The Switch-GRPO endpoint (star) sits at the high-accuracy end; the brevity-bonus operating point (diamond) trades a modest amount of accuracy for shorter outputs and zero max-length truncation. Dashed curve: empirical Pareto frontier.

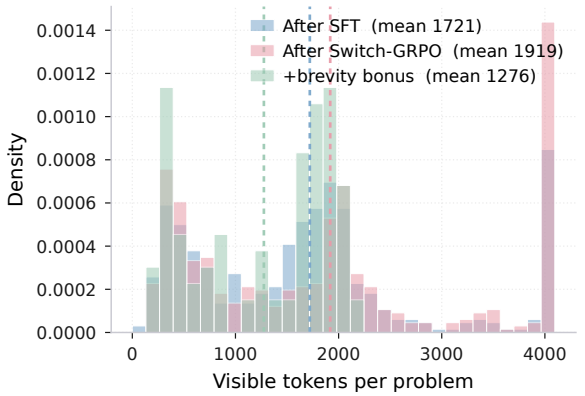

Figure 5: **Distribution of visible tokens per problem,** SFT vs. Switch-GRPO vs. +brevity. Dashed lines mark each checkpoint's mean.

SWITCH reaches **79.3**% on MATH-500 and **89.2**% on GSM8K, above all Coconut-style baselines under the matched-base-model protocol. The ablation, accuracy–efficiency and mechanistic analyses in §§4.2–5 are reported on a representative training run for which we have full per-step training, decoding and intervention logs.

What Does Switch-GRPO Add Over the Curriculum? Table 1 is the right number to report, but it does not show *where* the gain comes from. To separate the effect of RL from the effect of the curriculum alone, we evaluate the strongest curriculum-SFT checkpoint and the same checkpoint after Switch-GRPO on the same MATH-500 set with the same $K_{\min} = 4$ greedy decoding (Table 2).

Two observations matter. Latent-conditional accuracy jumps by $+\mathbf{12.6}$ points, attributable to RL alone since the underlying weights, vocabulary, and decoding path are identical. The switch rate drops from 81% to 58% at the same time. The model has not learned to invoke latent reasoning indiscriminately; it has learned to pick problems where the latent step pays off. Figure 3 shows the same calibration happening continuously over the training run: as RL proceeds, latent invocations per problem drop from $\sim$1.5 to $\sim$1 and visible-token usage contracts from $\sim$2900 to $\sim$1900. §5 returns to this calibration and shows that the underlying switch policy is sharpened, not erased.

A per-subject and per-difficulty breakdown (Appendix G) shows that the gain is spread broadly: SWITCH is strongest on algebraic and structurally regular subjects (Algebra 88.7%, Prealgebra 80.5%, Number Theory 79.0%) and accuracy degrades smoothly from 93.0% at level 1 to 53.7% at level 5, with no cliff.

An Accuracy–Efficiency Operating Curve So far we have reported one operating point. By varying the Switch-GRPO reward, in particular by adding a correctness-gated brevity bonus (Appendix A), the user can pick a point along an explicit accuracy–length curve (Fig. 4). The brevity-bonus operating point trades about three points of accuracy for $\sim$ 33% shorter outputs and 0% max-length truncation. The full visible token distribution (Fig. 5) shows that this is a distributional shift rather than just a change of mean: the brevity variant moves probability mass below the SFT median while losing very few problems to the high-token tail. The matched empirical CDF is in Appendix E.

Method	Reasoning style	MATH-500		GSM8K	
		Acc.	Tokens	Acc.	Tokens
Non-latent baselines					
No-CoT (direct answer)	—	11.3	14	28.4	12
Text-CoT (SFT)	explicit	80.6	2 079	88.6	1 819
iCoT (Deng et al., 2024)	internalised	24.8	9.6	60.4	9.5
Pause Tokens (Goyal et al., 2024)	non-decoding	14.6	14.7	33.7	13.5
Coconut-style latent CoT					
Coconut (Hao et al., 2025)	hidden-state recurrence	46.6	9.6	76.1	9.8
CODI (Shen et al., 2025)	hidden-state recurrence	48.3	10.2	76.4	9.9
CoLaR (Tan et al., 2025)	hidden-state recurrence	53.6	11.8	78.5	10.6
SWITCH (ours, after Switch-SFT)	hidden-state recurrence	66.7	1 433	80.2	1 249
SWITCH (ours, after Switch-GRPO)	hidden-state recurrence	**79.3**	**1 721**	**89.2**	**1 608**

Table 1: **Headline comparison on MATH-500 and GSM8K against Coconut-style latent reasoning baselines.** All methods share a common Qwen3-8B base model (Qwen Team, 2025) under matched training data and decoding settings; baseline numbers come from our own re-implementations (Appendix A). "Acc." is accuracy (%); "Tokens" is the average number of visible (text) tokens per problem.

Stage	Latent acc.	Switch %	Avg. tok.
After SFT (curriculum)	66.7	81	1 433
After Switch-GRPO	**79.3**	58	1 777

Table 2: Effect of Switch-GRPO on latent reasoning ability. "Latent acc." restricts accuracy to test problems on which the model emitted at least one <swi> block.

Event	H	p(<swi>)	rank	margin
After SFT				
SWI-START	0.203	0.847	1.13	+3.48
random	0.068	0.003	1003.9	−21.9
After Switch-GRPO				
SWI-START	0.532	0.480	1.68	+0.08
random	0.131	0.002	1127.9	−16.8

Table 3: **Teacher-forced switch statistics.** Annotated <swi> positions vs. random non-boundary positions, on both checkpoints.

5 How Does Latent Work in Reasoning?

The numbers in §4.2 say that Switch-GRPO produces a better model than the curriculum-only SFT baseline. They do not say *why*. We use the explicit <swi>/</swi> boundaries as anchors to look at the trained model and answer three questions in sequence: (Q1) does the model emit <swi> with the localized structure of a learned switching policy, (Q2) does the latent step that follows contribute causally to the answer, and (Q3) where inside the latent block does that contribution sit? We answer Q1 with three observations , Q2 with a causal intervention, and Q3 with two complementary probes in this section. Throughout we instrument two checkpoints from the same training run: the curriculum-SFT checkpoint (**After SFT**) and the post-RL endpoint (**After Switch-GRPO**).

<swi> Is a Sharply Localised Boundary Token We teacher-force the model on the prefix immediately before an annotated <swi> position and read out the next-token distribution, using random non-boundary positions as a control. At annotated <swi> positions, the model places <swi> essentially at the top of the vocabulary (rank ≤ 1.7 on both checkpoints); at random non-boundary positions, it suppresses <swi> by orders of magnitude (rank $\sim 10^3$, $p \sim 10^{-3}$). The contrast is roughly four orders of magnitude in p(<swi>) and three orders of magnitude in rank. <swi> is not a formatting artefact the model emits uniformly: it is a control action whose distribution cleanly separates reasoning boundaries from ordinary continuation positions (Table 3).

The Switch-Window Has a Clean Spike. The next question is whether the model emits <swi> only at the boundary, or whether it emits <swi> over a few-token region around it. Figure 6 plots p(<swi>) at relative offsets $-8, \ldots, +8$ around each annotated <swi> position. The spike at offset 0 is followed by a collapse of several orders of magnitude one token later, on both checkpoints; <swi> is a boundary token, not a stylistic tag spanning a window. The comparison between checkpoints also sharpens the calibration story of §4.2: after SFT the peak sits at p(<swi>) $= 0.85$ with large positive margin to the next token; after Switch-GRPO it softens to 0.48 with margin near zero, but

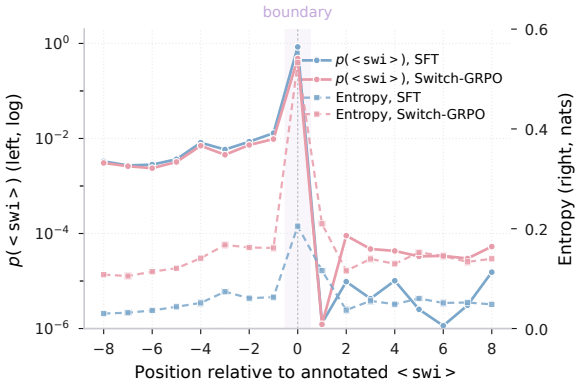

Figure 6: **Switch-window curves.** p(<swi>) (solid, left log axis) and entropy (dashed, right linear axis) at relative offsets $-8, \ldots, +8$ around annotated <swi> positions. The spike at the boundary collapses by several orders of magnitude one token later. Switch-GRPO preserves the spike but softens its peak height; per-offset values at $-1, 0, +1$ in Table 4.

Checkpoint	p_{-1}	p_0	p_{+1}
After SFT	1.3×10^{-2}	0.847	2×10^{-6}
After Switch-GRPO	9.7×10^{-3}	0.480	2×10^{-6}

Table 4: **Switch-window probabilities at the three central offsets** $k\in\{-1,0,+1\}$, with $p_k \equiv p$(<swi>).

the contrast to the immediate neighbours stays at $\sim 10^2$. RL has not erased the switch policy. It has made the model less aggressive at boundaries it is uncertain about, consistent with the halved switch rate and the $+12.6$-point jump in latent-conditional accuracy of Table 2.

Switch State Is Linearly Decodable From Late Layers. If the switch decision is being computed inside the model rather than memorized at the output, we should be able to read it out of the model's activations before the LM head. We fit a balanced ℓ_2-regularized logistic probe on the hidden state at seven layer offsets, with the binary label "next token is <swi>" (Fig. 7). The probe is near chance in the early layers, becomes moderately predictive in the middle, and reaches 91.9% at the last layer (After SFT) or 88.4% (After Switch-GRPO): the classic "feature emerges with depth" picture (Tenney et al., 2019; Belinkov, 2022). Switch-GRPO loses about three points at the last layer but keeps the early- and mid-layer profile, consistent with the softer-but-still-localized boundary policy.

Takeaway (Q1). <swi> *behaves as a learned switching policy, not a stylistic tag.*

Layer offset	After SFT	After Switch-GRPO
-24	0.533	0.537
-20	0.572	0.579
-16	0.579	0.576
-12	0.684	0.651
-8	0.746	0.748
-4	0.795	0.810
-1	**0.919**	**0.884**

Table 5: **Probe accuracy by layer offset** (balanced SWI-START vs. non-boundary classification).

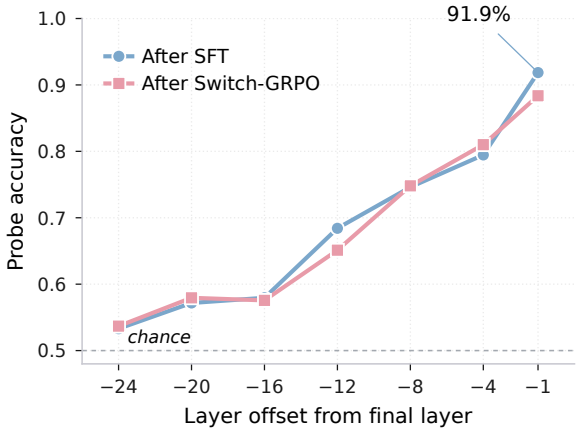

Figure 7: **Linear probe accuracy** for "next token is <swi>" at seven layer offsets. Both checkpoints rise from near chance in the early layers to $\geq 88\%$ at the last layer (exact numbers in Table 5).

The Latent Step Is Causally Doing Work. The three observations so far show that <swi> is a switching decision. They do not yet show that the latent step that follows is doing useful work. To test that, we intervene on the injected hidden state at every latent position (Meng et al., 2022; Heimersheim and Nanda, 2024) and compare four inference modes: NORMAL (default generation), ZERO (replace $\boldsymbol{h}_{t-1}$ in Eq. 1 with the zero vector), RANDOM-NORM (replace it with a random vector of the same norm), and SKIP (omit the latent step and generate as if <swi> had not been emitted). We run all four modes on MATH-500 and report two summaries (Fig. 8, Table 6): the unrestricted average and a diagnostic subset of problems where NORMAL both used latent reasoning and answered correctly. The diagnostic subset is the most informative for our question, because on those problems the latent step is the only thing that could matter.

The contrast in the diagnostic subset is unambiguous. Zeroing the latent state collapses accuracy from 100% to 33.3%; replacing it with a random vector of the same norm costs only 9.5 points; skip-

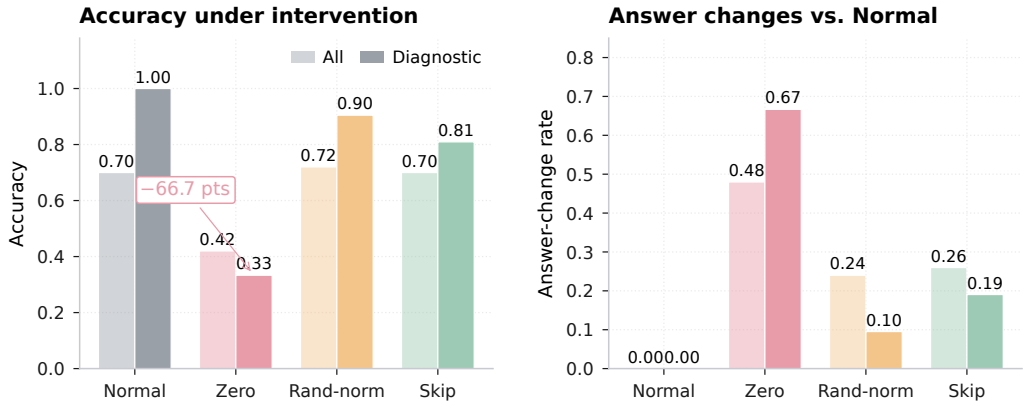

Figure 8: **Latent-state intervention.** Accuracy (left) and answer-change rate (right) under four intervention modes on the unrestricted MATH-500 set (light bars) and on the diagnostic subset (dark bars). ZERO collapses diagnostic accuracy from 100% to 33% (−66.7 points); RANDOM-NORM and SKIP are far less destructive. The latent computation is the specific hidden state of Eq. 1, not just any non-zero perturbation.

	All		Diagnostic	
Mode	Acc.	Δ ans.	Acc.	Δ corr.
NORMAL	0.70	0.00	1.000	0.000
ZERO	0.42	0.48	0.333	**−0.667**
RANDOM-NORM	0.72	0.24	0.905	−0.095
SKIP	0.70	0.26	0.810	−0.190

Table 6: **Latent-state intervention numbers.** "Δ ans." is the fraction of problems whose extracted answer changes vs. NORMAL; "Δ corr." is the change in correctness on the diagnostic subset.

Group	step 1	step 2	step 3	step 4
correct	0.9998	1.0000	0.9904	1.0000
wrong	1.0000	1.0000	0.9951	0.9999

Table 7: **Exit probability** p(`</swi>`) **inside latent,** stratified by whether the rollout is eventually correct. The model is ready to exit immediately after entering.

ping the latent step costs 19.0 points. Two conclusions follow. The latent computation is not just *any* non-zero perturbation, since a same-norm random vector is nearly harmless; and the latent step is not redundant text in disguise, since skipping it costs twice as much as random noise.

Takeaway (Q2). *The latent step performs causally important computation, not a generic perturbation or redundant text in disguise.*

Where the Latent Computation Lives. Q2 established that the latent step matters causally; we now ask *where* in the block the work happens. Two probes converge on an answer. First, the logit lens (nostalgebraist, 2020; Belrose et al., 2023), applied via $\mathrm{softmax}(W\boldsymbol{h}_t)$, returns `</swi>` as the top-1 token at every step, but at the *first* latent step the top-k becomes more diffuse and problem-conditional (e.g., `inverse`, `arc`, `angle` on a trigonometry problem). Second, p(`</swi>`) ≈ 1 at every latent step regardless of correctness (Table 7); without the $K_{\min}$ constraint, the latent block would collapse to a single hidden forward pass.

Together, the latent block reduces to a single hidden-state transition on entry, kept from collapsing by $K_{\min}$ while the curriculum's fixed-offset `</swi>` makes the remainder exit-ready. This also explains the Q2 ordering: zeroing destroys the transition that carries the reasoning, a same-norm random vector preserves local geometry, and a `<latent>`-skip walks past it.

Takeaway (Q3). *The work in the latent block is concentrated at a single hidden-state transition on entry, kept alive by the $K_{\min}$ constraint.*

6 Conclusion

We presented SWITCH, a switchable latent reasoning framework integrating a learned switch token, a three-phase curriculum, and a Switch-GRPO optimizer into hidden-state-injection models. Extensive experiments show that SWITCH outperforms competitive baselines, while providing an adaptable accuracy–efficiency trade-off. Furthermore, its explicit boundary design allows for rigorous verification: the switch decision is highly localized and linearly decodable (91.9% probe accuracy), and causal analysis confirms the functional necessity of the latent reasoning steps. Overall, our work proves that recurrent latent CoT can be successfully optimized via RL and directly interpreted.

Limitations

Our experiments are restricted to 8B-parameter Qwen3 models and to mathematical reasoning benchmarks (MATH-500 and GSM8K). We have not yet evaluated SWITCH on multi-domain reasoning or at larger model scales, where the right balance between learned switching and latent depth may differ. Switch-GRPO's gradient flows through the text segments of each rollout only; latent positions contribute via a frozen KV cache (Appendix A), so the latent representation itself is shaped primarily by the Phase 2 curriculum rather than directly by the RL objective. Our mechanistic analysis is oriented toward what is encoded by the model (switch boundaries, latent causal effect) rather than at characterising failure modes; the logit-lens decoding in §5 is qualitative and should not be read as a faithful reconstruction of the latent reasoning trajectory. A combined system in which the latent token itself is also samplable, bridging hidden-state recurrence and vocabulary-mixture latents, is a natural next step, and a head-to-head comparison at matched scale and data is left to future work.

References

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. In *arXiv preprint*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2024. Do NOT think that much for 2+3=? on the overthinking of o1-like LLMs. *Preprint*, arXiv:2412.21187.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. In *arXiv preprint*.

DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jingcheng Deng, Liang Pang, Zihao Wei, Shicheng Xu, Zenghao Duan, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. 2025. LLM latent reasoning as chain of superposition. *Preprint*, arXiv:2510.15522.

Jingcheng Deng, Zihao Wei, Liang Pang, Junhong Wu, Shicheng Xu, Zenghao Duan, and Huawei Shen. 2026. Latent-GRPO: Group relative policy optimization for latent reasoning. *Preprint*, arXiv:2604.27998.

Yuntian Deng, Yejin Choi, and Stuart Shieber. 2024. From explicit CoT to implicit CoT: Learning to internalize CoT step by step. In *arXiv preprint*.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. Think before you speak: Training language models with pause tokens. In *International Conference on Learning Representations (ICLR)*.

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 *Conference on Language Modeling (COLM)*.

Stefan Heimersheim and Neel Nanda. 2024. How to use and interpret activation patching. *Preprint*, arXiv:2404.15255.

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 *NeurIPS Datasets and Benchmarks Track*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*.

Hugging Face. 2025a. Math-verify: A robust mathematical expression evaluator.

Hugging Face. 2025b. OpenR1-Math-220k.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*.

nostalgebraist. 2020. Interpreting GPT: The logit lens. LessWrong post.

OpenAI. 2024. Learning to reason with LLMs. OpenAI blog post, September 12, 2024.

Jacob Pfau, William Merrill, and Samuel R. Bowman. 2024. Let's think dot by dot: Hidden computation in transformer language models. *Preprint*, arXiv:2404.15758.

Qwen Team. 2025. Qwen3: Think deeper, act faster.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, 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. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

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 (EMNLP)*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. HybridFlow: A flexible and efficient RLHF framework. *Preprint*, arXiv:2409.19256.

Dachuan Shi, Abedelkadir Asi, Keying Li, Xiangchi Yuan, Leyan Pan, Wenke Lee, and Wen Xiao. 2026. SwiReasoning: Switch-thinking in latent and explicit for Pareto-superior reasoning LLMs. In *International Conference on Learning Representations (ICLR)*.

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

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Jian Luan, and Ruihua Song. 2025. Think silently, think fast: Dynamic latent compression of LLM reasoning chains. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. 2025. Soft thinking: Unlocking the reasoning potential of LLMs in continuous concept space. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhi Zheng, Yu Gu, Wei Liu, Yee Whye Teh, and Wee Sun Lee. 2025. SofT-GRPO: Surpassing discrete-token LLM reinforcement learning via gumbel-reparameterized soft-thinking policy optimization. *Preprint*, arXiv:2511.06411.

A Implementation Details

Special tokens. We register three special tokens with IDs 151 669 (`<swi>`), 151 670 (`</swi>`), and 151 671 (`<latent>`), resizing the Qwen3-8B input/output embeddings from 151 936 to 151 672. The new embeddings are initialised by anchoring to a content-neutral seed token, which we found necessary to avoid rank-collapse with $\mathcal{N}(0, \sigma)$ initialisation at 8B scale.

Phase-1 SFT. We train Phase 1 with LoRA (rank 32, $\alpha = 64$; Hu et al., 2022) on all $\{q, k, v, o, \text{gate}, \text{up}, \text{down}\}$ projections, together with the resized embeddings and the LM head. The standard supervised cross-entropy loss reaches 0.098 on the annotated OpenR1-Math corpus.

Phase-2 curriculum. We use $c = 2$ and $K_{\max} = 8$ (up to 16 `<latent>` tokens per span), a per-sample latent cap of 48 to avoid OOM on samples with many spans, and a curriculum-stage smoothing probability $p_{\text{unif}} = 0.1$. We sweep $k \in \{0, \ldots, 8\}$ for three epochs per stage, warm-starting each stage from the previous checkpoint. The parallel replacement strategy is our default and underlies all Phase-3 initialisations; the sequential warm-up restricts the replacement at stage k to the leftmost k spans and is reported in §I.

Hardware. We train and evaluate on a single node of 8×NVIDIA H20 GPUs (95 GB each) under PyTorch DDP. Switch-GRPO processes one question per GPU with $G = 5$ rollouts per question, applying gradient updates atomically. We deliberately avoid vLLM-style text-only rollout (Sheng et al., 2024) because it bypasses hidden-state injection and breaks training and evaluation alignment for latent reasoning.

Switch-GRPO hyperparameters. Group size $G = 5$, clip threshold $\varepsilon_c = 0.2$, KL coefficient $\beta = 10^{-3}$, learning rate 10^{-6}, three inner epochs per rollout. We use $\pi_{\theta_{\text{old}}}$ as the KL anchor in place of a separate reference model, saving roughly 18 GB/GPU. Rollouts use temperature 0.5 in the main configuration and 0.7 in the compression-oriented operating point of §4.2, $K_{\min} = 4$ in the main configuration and $K_{\min} = 2$ in the compression operating point, and max_new_tokens $\in \{2048, 4096, 6000\}$ depending on the configuration.

Segmented backward. Storing one autograd graph for G rollouts, each containing many latent

passes interleaved with text, does not fit in high-bandwidth GPU memory (HBM) at 8B scale. We split every rollout at <swi>/</swi> boundaries and process the segments left-to-right, with a streaming key-value cache passing between them. Latent segments run inside `torch.no_grad()` and store nothing; text segments run with gradient enabled, contribute their term to the clipped surrogate loss of Eq. 9 (below), and call `backward()` immediately. Peak activation memory becomes that of one text segment, so the only price we pay is that the latent positions contribute to the gradient pass only through the KV cache they hand off to the next text segment.

Reported metrics. For every checkpoint we report overall accuracy on each benchmark, the switch rate (the fraction of test problems on which the model emits at least one <swi> block), and the average visible-token count per problem. Latent forward passes are not counted as visible tokens.

Compression-oriented operating point. For the shorter-output operating point of §4.2 we extend Switch-GRPO from the default checkpoint with an additional brevity component $w_b\, r_{\text{brev}}$ in the reward (Eq. 6), $w_b = 0.1$, $T_{\text{lo}} = 800$, $T_{\text{hi}} = 2000$. The brevity bonus is gated on *correct* responses that used at least one <swi> block, so the model is never rewarded for producing short wrong answers. This operating point reaches 69.0% MATH-500 accuracy at $1\,276$ average visible tokens with 0% max-length truncation, versus the default 72.6% at $1\,919$ tokens with 18.4% truncation, a controllable Pareto trade-off rather than a separate "best" system.

Baseline re-implementations. For Table 1 we re-implement every comparison method on the same Qwen3-8B base model and OpenR1-Math training corpus. The *no-CoT* baseline emits only the final answer; the *text-CoT SFT* baseline trains the model on full visible CoT without any switch token; *iCoT* (Deng et al., 2024) progressively internalises CoT steps; *Pause Tokens* (Goyal et al., 2024) insert non-decoding <pause> tokens before the answer. For Coconut (Hao et al., 2025), CODI (Shen et al., 2025), and CoLaR (Tan et al., 2025) we follow each paper's reference recipe but on Qwen3-8B: Coconut uses the multi-stage curriculum with $c = 2$ latent positions per text token; CODI uses the single-stage self-distillation objective with matched teacher/student; CoLaR uses a separate "latent head" that predicts compressed embeddings.

Robustness. We wrap rollout and the gradient pass in `try/except OutOfMemoryError` and propagate a `survived_indices` mask through the group-relative advantage (Eq. 7 below), so a single long rollout does not corrupt the advantage of an entire batch. Each question's G rollouts are processed atomically (no gradient accumulation across questions).

B Switch-GRPO Loss, in Full

Reward components. The reward terms described in §3.3 of the main body are defined as follows. Let $\hat{y}$ be the answer extracted from o via standard `\boxed{}` parsing, $\equiv$ denote mathematical equivalence judged by `math-verify` (Hugging Face, 2025a), $\mathbf{1}_{\text{wf}}$ indicate well-formed <swi>/</swi> tags, $\mathbf{1}_{\text{used}}$ indicate that at least one <swi> block is present, and $|o|$ the number of visible tokens:

$$r_{\text{corr}} = 2\,\mathbf{1}[\hat{y} \equiv y^{\star}] - 1, \tag{3}$$

$$r_{\text{fmt}} = 2\,\mathbf{1}_{\text{wf}} - 1, \tag{4}$$

$$r_{\text{use}} = r_{\text{corr}} \cdot \mathbf{1}_{\text{wf}} \cdot \mathbf{1}_{\text{used}}, \tag{5}$$

$$r_{\text{brev}} = \text{clip}\Big(\tfrac{T_{\text{hi}} - |o|}{T_{\text{hi}} - T_{\text{lo}}}, 0, 1\Big) \cdot \mathbf{1}[\hat{y} \equiv y^{\star}]\, \mathbf{1}_{\text{used}}, \tag{6}$$

with $T_{\text{lo}} = 800$ and $T_{\text{hi}} = 2000$. The brevity bonus is gated on correct responses that used <swi>, so the model is never rewarded for producing short wrong answers.

Group-relative advantages. Following Shao et al. (2024) we use a trajectory-level advantage shared across the rollout's text-positions,

$$A^{(i)} = \frac{R^{(i)} - \mu_R}{\sigma_R + \varepsilon}, \tag{7}$$

where μ_R and σ_R are the mean and standard deviation of $\{R^{(j)}\}_{j=1}^{G}$ and $\varepsilon = 10^{-8}$.

Policy ratio. The per-text-position policy ratio at the frozen rollout prefix is

$$\rho_t^{(i)}(\theta) = \frac{\pi_\theta(x_t^{(i)} \mid \tilde{\boldsymbol{e}}_{<t}^{(i)})}{\pi_{\theta_{\text{old}}}(x_t^{(i)} \mid \tilde{\boldsymbol{e}}_{<t}^{(i)})}, \tag{8}$$

for $t \in \mathcal{T}_{\text{text}}^{(i)}$. The conditioning $\tilde{\boldsymbol{e}}_{<t}^{(i)}$ is frozen from the rollout: we replay the identical input-embedding sequence at the gradient pass, so the same hidden-state injections that produced the reward are the ones that backpropagate through the surrounding text.

Clipped surrogate loss. The full Switch-GRPO objective is the standard PPO-style clipped surrogate plus a KL penalty, summed over text-positions of all rollouts:

$$\mathcal{L}_3 \;=\; -\,\mathbb{E}_q \sum_{i,\,t\in\mathcal{T}_{\text{text}}^{(i)}} \Big[\, L_t^{(i)} - \beta\, \widehat{\mathrm{KL}}_t^{(i)} \,\Big] \Big/ N_{\text{tok}}, \qquad (9)$$

with $L_t^{(i)} = \min\big(\rho_t^{(i)} A^{(i)},\, \mathrm{clip}(\rho_t^{(i)}, 1{-}\varepsilon_{\text{c}}, 1{+}\varepsilon_{\text{c}}) A^{(i)}\big)$, $N_{\text{tok}} = \sum_i |\mathcal{T}_{\text{text}}^{(i)}|$, $\varepsilon_{\text{c}} = 0.2$, $\beta = 10^{-3}$, and the unbiased KL estimator $\widehat{\mathrm{KL}}_t^{(i)} = \rho_t^{(i)} - 1 - \log \rho_t^{(i)}$ (Schulman et al., 2017). We use $\pi_{\theta_{\text{old}}}$ both as the importance-sampling reference and as the KL anchor, removing the need for a separate frozen reference model.

C Per-Checkpoint Trajectory of SWITCH

The trajectory we use throughout the main body for ablation (§4.2), accuracy–efficiency (§4.2), and mechanistic (§5) analyses is taken from a representative training run for which we have full per-step training, decoding, and intervention logs. Table 8 reports its MATH-500 numbers across the curriculum-SFT stages and Switch-GRPO optimizer steps. The headline numbers reported in Table 1 (79.3% MATH-500, 89.2% GSM8K) come from our strongest end-to-end Switch-GRPO run produced by the same pipeline; the trajectory in Table 8 provides the matched diagnostic context.

Full training trajectory. Figure 9 extends the main-body Fig. 3 from step 1,000 to the entire 1,964-step run, including the late-training region (shaded) in which the policy enters a reward-hacking regime: switch rate climbs to 100%, latent invocations per problem explode from $\sim$1 to $\sim$13, and average reward drifts downward as the model invokes <swi> without converting the extra latent compute into correct answers. We early-stop at step 800, the operating point reported throughout the main body; in the appendix view the dashed vertical line and the shaded region together justify this choice.

D Algorithm Boxes

E Visible-Token CDF

Figure 10 reports the empirical CDF of visible tokens per problem for the SFT baseline, the Switch-GRPO endpoint, and the brevity-bonus operating point on MATH-500, complementing the main-body histogram (Fig. 5). The brevity-bonus variant dominates the SFT baseline up to the median while losing very few problems to the high-token tail.

Checkpoint	$K_{\min}$	Acc.	Lat. acc.	Swi %	Tok.
After SFT (curriculum only)					
stage 8	0	53.0	46.1	80	1721
stage 6	4	70.0	66.7	81	1433
After Switch-GRPO					
step 200	4	78.0	74.6	67	1609
step 600	4	75.0	66.1	62	1753
step 800	4	**72.6**	**67.8**	58	1919
+ *brevity bonus*					
step 1000	4	69.0	75.4	57	1276

Table 8: **Full per-checkpoint MATH-500 trajectory of the representative SWITCH training run.** "After SFT" refers to curriculum-only checkpoints (no RL); "After Switch-GRPO" refers to RL post-training from the strongest SFT checkpoint with the default correctness + format + latent-usage reward; "+ brevity bonus" adds the compression-oriented brevity reward term (§A). The bolded row is the post-RL endpoint of this representative run; the strongest end-to-end Switch-GRPO run produced by the same pipeline reaches 79.3% MATH-500 / 89.2% GSM8K (Table 1).

Algorithm 1 `CoconutSwiModel` forward.

Require: tokens $x_{1:T}$, latent mask $L \in \{0,1\}^T$
Ensure: hidden states $\boldsymbol{h}_{1:T}$, logits $\boldsymbol{\ell}_{1:T}$
1: partition $1 : T$ into maximal constant-L segments $S_1, \ldots, S_M$
2: $\mathcal{K} \leftarrow \emptyset$ ▷ streaming KV cache
3: **for** $m = 1, \ldots, M$ **do**
4: $\quad \boldsymbol{e}_t \leftarrow E[x_t]$ if $L_t = 0$ else $\boldsymbol{h}_{t-1},\ t \in S_m$
5: $\quad (\boldsymbol{h}_{S_m}, \mathcal{K}) \leftarrow f_\theta(\boldsymbol{e}_{S_m}; \mathcal{K});\ \ \boldsymbol{\ell}_{S_m} \leftarrow W \boldsymbol{h}_{S_m}$
6: **end for**
7: **return** $\boldsymbol{h}_{1:T}$, $\boldsymbol{\ell}_{1:T}$

F Mechanistic Analysis: Additional Details

Probe details. The probe in Table 5 of the main body is a balanced ℓ_2-regularised logistic classifier ($C = 1.0$) with an 80 : 20 train/test split. Reported numbers are test-set accuracies.

Teacher-forced switch metrics. The numbers in Table 3 are computed by teacher-forcing each instance on the prefix immediately before an annotated <swi> position, sampling random non-boundary positions per checkpoint as a negative control, and reading out the entropy, $p($<swi>$)$, the rank of <swi>, and the log-margin to the top token. We use the same MATH-500 evaluation problems for both checkpoints so the contrasts are paired.

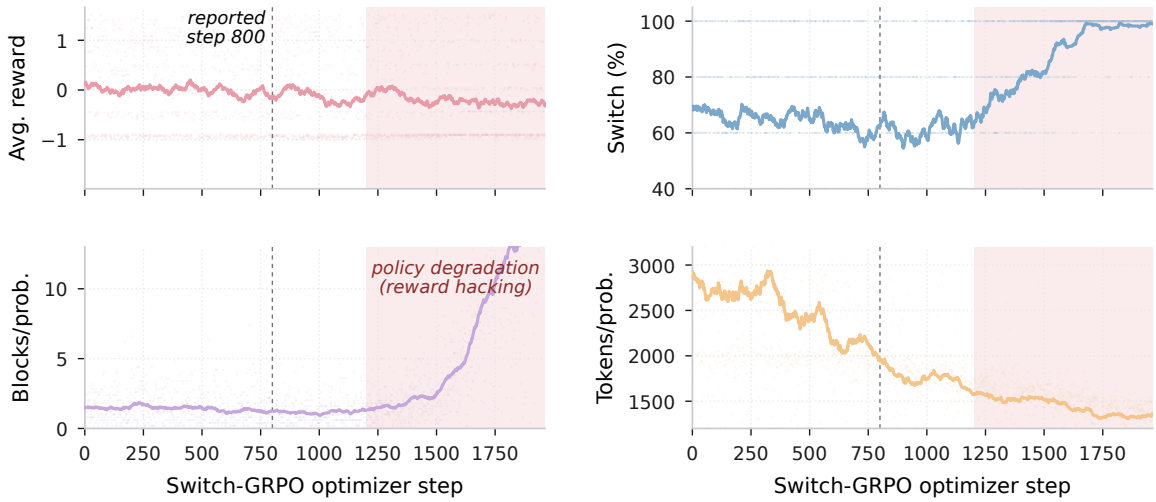

Figure 9: **Full Switch-GRPO trajectory** (1,964 optimizer steps) on the representative training run. Light scatter is raw per-batch metrics; coloured lines are 80-step rolling means. The pink-shaded region marks the late-training reward-hacking regime ($\gtrsim$step 1,200); the dashed vertical line marks the reported step-800 checkpoint, our early-stopping point. The main-body Fig. 3 shows the same panels cropped to the stable 0–1,000 window.

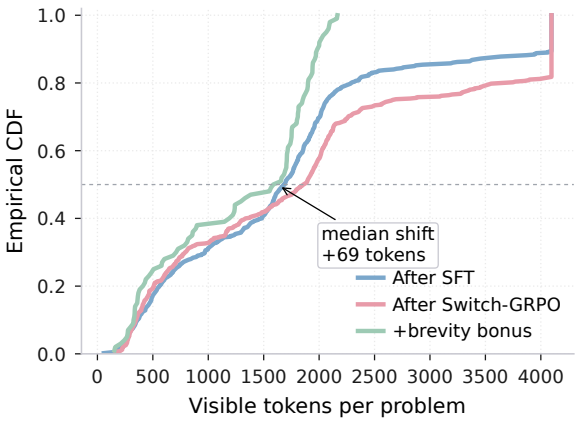

Figure 10: **Empirical CDF of visible tokens per problem.**

Switch-window window size. Figure 6 uses offsets $-8, \ldots, +8$ around each annotated <swi> position. The collapse one token after the boundary ($\sim 2 \times 10^{-6}$ for p(<swi>), rank $\sim 5\,000$) is robust to window-size choice.

Probe details. We balance the binary classification dataset by sampling an equal number of non-boundary positions per SWI-START position, and fit a logistic probe with default ℓ_2 regularisation ($C = 1.0$) and a single $80:20$ train/test split. Reported numbers are test-set accuracies.

Intervention modes. For the diagnostic subset of Table 6 we first run NORMAL inference and restrict to the problems where it (i) produced at least one latent block and (ii) was graded correct by `math-verify`. Each intervention mode is then run under greedy decoding with $K_{\min} = 4$ and the same max_new_tokens as the headline evaluation.

G Per-Subject and Per-Difficulty Visualisation

Table 9 reports SWITCH's MATH-500 accuracy split by subject and by difficulty level for the headline post-RL checkpoint. Figure 11 additionally splits each subject into "with latent" and "without latent" branches.

Subject	Acc. (%)
Algebra	88.7
Prealgebra	80.5
Number Theory	79.0
Precalculus	67.9
Counting & Probability	60.5
Intermediate Algebra	56.7
Geometry	53.7
Level 1	93.0
Level 2	90.0
Level 3	83.8
Level 4	64.1
Level 5	53.7

Table 9: **Per-subject (top) and per-difficulty (bottom) MATH-500 accuracy** of SWITCH after Switch-GRPO on the representative training run.

H Generation Trace Analysis

Failure profile of wrong trajectories. We additionally stratify SWITCH's post-RL trajectories by

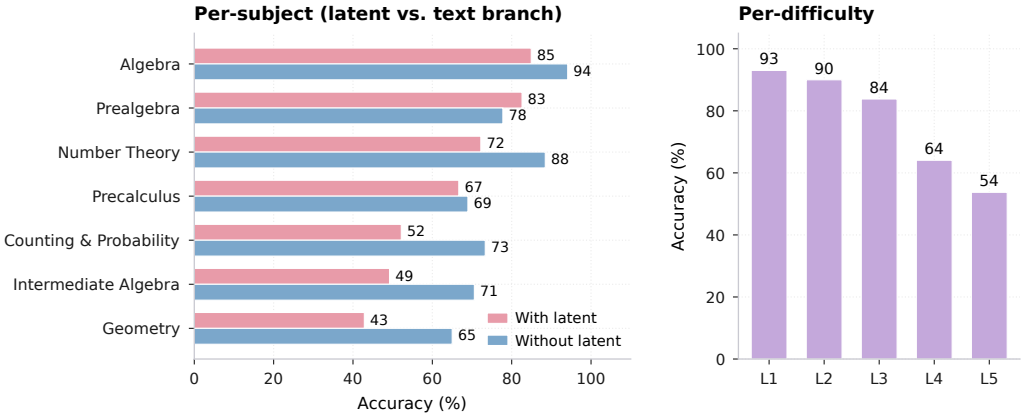

Figure 11: **Per-subject (left) and per-difficulty (right) MATH-500 accuracy** of SWITCH after Switch-GRPO.

Algorithm 2 One Switch-GRPO step.

Require: prompt q, gold $y^\star$, group size G, clip ε_c, KL coef β, min latent dwell $K_{\min}$

// ROLLOUT (NO_GRAD), REAL HIDDEN-STATE INJECTION
for $i = 1, \ldots, G$ **do**
  $o^{(i)} \leftarrow$ GENERATE_RL(q, θ_{old}, $K_{\min}$) ▷ Eq. 1
  store $\mathcal{T}^{(i)}_{\text{text}}$, $\tilde{\boldsymbol{e}}^{(i)}_{1:T_i}$
  $R^{(i)} \leftarrow R(o^{(i)}, q, y^\star)$ ▷ Eq. 6
end for
$\{A^{(i)}\}_{i=1}^{G} \leftarrow \text{normalise}\{R^{(i)}\}$ ▷ Eq. 7
// SEGMENTED BACKWARD, GRADIENT THROUGH TEXT ONLY
for $i = 1, \ldots, G$ **do**
  partition $o^{(i)}$ into segments $S^{(i)}_{1:M_i}$ at <swi>/</swi>
  $\mathcal{K} \leftarrow \emptyset$; $\ell_{\text{cum}} \leftarrow 0$
  for $m = 1, \ldots, M_i$ **do**
    if $S^{(i)}_m$ is a latent segment **then**
      run f_θ on $S^{(i)}_m$ in no_grad, update $\mathcal{K}$
    else ▷ text segment, gradient on
      compute $\rho^{(i)}_t$ on $S^{(i)}_m$ ▷ Eq. 8
      $\ell_{\text{seg}} \leftarrow$ contribution of $S^{(i)}_m$ to $\mathcal{L}_3$ ▷ Eq. 9
      ℓ_{seg}.BACKWARD(); $\ell_{\text{cum}} \mathrel{+}= \ell_{\text{seg}}$.detach
      update $\mathcal{K}$ with this segment's (K, V)
    end if
  end for
end for
$\theta \leftarrow$ OPTIM.STEP(); $\theta_{\text{old}} \leftarrow \theta$
return $\ell_{\text{cum}}/N_{\text{tok}}$, $\{R^{(i)}\}$

correctness and by latent usage on MATH-500 (Table 10). Wrong trajectories are substantially longer than correct ones, both with and without latent usage, and their switch decisions are slightly less confident (entropy 0.717 vs. 0.608, p(<swi>) $= 0.669$ vs. 0.763).

I Ablations

Minimum latent dwell $K_{\min}$. We swept $K_{\min} \in \{0, 2, 4, 8, 16\}$ during inference; the $K_{\min} = 0$ setting collapses latent blocks to a single hidden forward pass and reduces accuracy to 53.0% (the curriculum-SFT MATH-500 number reported in the trajectory of Table 8), while $K_{\min} = 4$ recovers the training distribution and is our default (§5 gives the mechanistic justification).

Group	avg. tok.	avg. <swi>	avg. latent steps
correct, no <swi>	959	0.0	0.0
correct, with <swi>	1197	1.86	7.43
wrong, no <swi>	1729	0.0	0.0
wrong, with <swi>	1805	1.40	5.60

Table 10: **Generation trace analysis** of SWITCH (after Switch-GRPO) on MATH-500, $K_{\min} = 4$. Wrong trajectories are substantially longer than correct ones in both branches, consistent with the calibration story of §5.

J Full Related Work

We give the more detailed treatment of related work promised in §2.

J.1 Latent Chain-of-Thought Reasoning

Hidden-state recurrence (Coconut, CODI). Coconut (Hao et al., 2025) formulates continuous CoT by feeding the previous step's last-layer hidden state back as the next input embedding, so that an entire reasoning step happens in latent space between two text tokens. The model is trained with a multi-stage curriculum that progressively replaces explicit CoT tokens with $k \cdot c$ latent positions. CODI (Shen et al., 2025) keeps the same hidden-state-injection mechanism but replaces the curriculum with a single-stage self-distillation objective: a teacher path with full explicit CoT and a student path with a few continuous thoughts share weights, and the student's hidden state at the answer-adjacent token is aligned to the teacher's via an L_1 feature loss. Both methods inherit the

same latent geometry: each latent token has input embedding equal to a previous-step hidden state, a deterministic function of the input prefix. SWITCH continues this line.

Vocabulary-embedding mixtures. A more recent line redefines the latent token as a *convex combination* of vocabulary input embeddings. Soft-Thinking (Zhang et al., 2025) uses the next-token softmax probabilities as mixture weights, $s_t = \sum_{v \in \mathcal{V}} p_t(v)\, E[v]$, so s_t lies on the convex hull of the vocabulary embeddings. Latent-SFT (Deng et al., 2025) restricts this to a top-k mixture and trains with stochastic Gumbel-Softmax targets, reporting $2.7\times$–$5.5\times$ shorter chains than explicit SFT on six math benchmarks. Latent-GRPO (Deng et al., 2026) explicitly contrasts itself with Coconut, calling the latter "early methods which directly adopt the hidden state as the latent token," and proposes vocabulary-superposition with one-sided Gumbel margins as a more RL-friendly alternative. SofT-GRPO (Zheng et al., 2025) similarly adds Gumbel noise to logits to make Soft-Thinking RL-trainable. The shared property of vocabulary mixtures is that latent tokens are *samplable* via Gumbel-Softmax and have a tractable density, which is precisely what makes GRPO directly applicable. We do not include the vocabulary-mixture line in our headline comparison because it uses a different latent representation; its results are not directly comparable, and SWITCH's contribution is orthogonal: we show that the original hidden-state-recurrence representation can itself be made RL-trainable.

Scale. Prior hidden-state-recurrence work has operated mainly at GPT-2 / 1–2B scale: Coconut's main experiments use GPT-2 (Hao et al., 2025), and CODI uses GPT-2 and LLaMA-1B (Shen et al., 2025). The Coconut paper reports a brief LLaMA-3-8B probe in its appendix (improving GSM8K by 1.4 points over the SFT baseline), but without a tuned curriculum, a learned switching token, or RL, which is the regime SWITCH targets. The closely related CoLaR (Tan et al., 2025) is also hidden-state-recurrence-based but introduces a separate "latent head" that predicts compressed embeddings.

Other compression approaches. A simpler line inserts non-decoding "thinking" tokens without continuous-state feedback: pause tokens (Goyal et al., 2024), filler tokens (Pfau et al., 2024), and implicit-CoT internalisation (Deng et al., 2024). These do not maintain explicit text reasoning at all, whereas SWITCH preserves text CoT outside `<swi>` blocks so the visible trajectory remains interpretable.

J.2 Switchable / Hybrid Reasoning

The closest single work to ours is SwiReasoning (Shi et al., 2026). It takes a frozen reasoning LLM and dynamically switches between explicit decoding and a latent step based on the entropy trend of the next-token distribution. A hard *switch budget* caps the number of mode changes per problem (Chen et al., 2024; Snell et al., 2024), yielding 1.8–3.1 accuracy points and 57–79% token-efficiency gains on math, STEM, coding, and general benchmarks.

Why we still train. SwiReasoning is training-free, so the latent step is performed by a model that was not trained for it, and the location of switches is fixed by an external entropy rule. SWITCH instead *learns* a discrete switching token (`<swi>`) and the latent dynamics inside it, so both the entry point and the dwell of latent reasoning are optimised end-to-end, in particular by RL in Phase 3.

Adaptive test-time compute. A separate line trains LLMs to spend test-time compute adaptively (Snell et al., 2024; Chen et al., 2024), but always emits the extra "thinking" as text. SWITCH brings the adaptive-compute decision and the latent representation into a single trained model.

J.3 Reinforcement Learning for Reasoning and Latents

Group Relative Policy Optimization (GRPO; Shao et al., 2024) is the de-facto policy optimizer for post-training reasoning LLMs and underpins DeepSeek-R1 (DeepSeek-AI, 2025); Schulman et al. (2017) provides the PPO foundation.

RL for latent representations. Standard GRPO assumes the policy outputs a categorical distribution over discrete tokens and that rollouts sample from it. Vocabulary-mixture latents satisfy this through Gumbel-Softmax: Latent-GRPO (Deng et al., 2026) uses top-k vocabulary mixtures with one-sided Gumbel margins, invalid-sample advantage masking, and optimal-correct-path first-token selection; SofT-GRPO (Zheng et al., 2025) adds Gumbel noise to logits and uses Gumbel reparameterisation to assign credit to the underlying logits. Both methods rely on the latent being samplable *by*

construction: their RL story requires a tractable density at every latent position. Hidden-state-recurrence latents admit no such density. Switch-GRPO (§3.3) extends GRPO to this regime by defining the policy ratio only at text positions while keeping rollouts that perform real hidden-state injection. This complements the vocabulary-mixture line: we ask how far hidden-state recurrence can be pushed, given that it is weight-sharing with the LM and admits the simplest implementation.

J.4 Interpretability of Internal Reasoning States

A central concern with any latent-CoT method is whether the latent states carry meaningful reasoning content. **Logit-lens** analysis (nostalgebraist, 2020; Belrose et al., 2023) reads intermediate hidden states through the LM head to obtain a distribution over the vocabulary, offering a qualitative view of what the model "believes" at each layer or step. **Linear probing** (Tenney et al., 2019; Belinkov, 2022) trains a linear classifier on frozen activations to test whether a target property is encoded in a particular layer. **Causal activation interventions** (Meng et al., 2022; Heimersheim and Nanda, 2024) perturb specific activations and measure the effect on the output distribution, turning correlational evidence into a causal claim. We use all three in §5; to our knowledge this is the first study to apply this triad to a learned latent-CoT model at 8B scale.

K Extended Discussion

Hidden-state recurrence is RL-compatible. Recent latent-CoT work has argued that hidden-state-recurrence latents cannot be optimised with on-policy RL and has switched to samplable vocabulary mixtures (Deng et al., 2026; Zheng et al., 2025; Zhang et al., 2025; Deng et al., 2025). Switch-GRPO is a constructive counterexample. The key observation is that the GRPO policy ratio only needs a tractable density at the *decision points* of the policy, which are the text-positions that emit `<swi>`, `</swi>` and the visible answer. Latent positions have deterministic embeddings given the preceding text; they only need to be replayed identically at the gradient pass. Combined with engineering for the multi-pass forward (Appendix A), this lets us keep Coconut's latent formulation while still getting the gradient signal of GRPO.

What RL actually changes. The mechanistic analysis (§5) shows that Switch-GRPO does not erase the switch policy curriculum SFT already learned. After SFT, $p(\texttt{<swi>})$ is 0.85 at every annotated boundary with very low entropy; after Switch-GRPO it drops to 0.48 with entropy ~ 0.5, while the contrast to the immediate neighbours stays at $\sim 10^2$. The switch rate halves and latent-conditional accuracy nearly doubles, so the simplest reading, supported by the intervention result, is that RL has shifted probability mass away from boundaries where opening a latent block would not have helped and toward boundaries where it does.

The latent step is not a "black box". A persistent worry about non-decoding "thinking" tokens (Pfau et al., 2024; Goyal et al., 2024) is that they function as inert compute rather than as task-relevant reasoning steps. §5 addresses this for SWITCH's latent blocks: on problems where the model uses latent reasoning and answers correctly, zeroing the injected hidden states costs 66.7 accuracy points, while replacing them with a random vector of the same norm costs only 9.5 points. An arbitrary non-zero perturbation does not reproduce the latent computation; the specific hidden state Eq. 1 produces is what the answer depends on.