# Local Residual Controllers for Thinking-Mode Entry in Open Language Models

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Several open language models now expose public controls that select between thinking-style and direct-answer behavior. This paper studies whether such controls induce a recoverable internal branch state at the prompt boundary, before long-form generation begins. We introduce a prompt-boundary branch-state audit. Given paired public prompts in two modes, we learn a model-specific width-2 residual-tail controller, freeze it, and evaluate whether it steers held-out direct/off prompt-boundary states toward the paired thinking-branch next-token distribution without using held-out donor activations, generated answers, or correctness labels. In the same-checkpoint public-switch settings studied here, Qwen3.5-4B, Gemma 4 E2B-it, and Nemotron Nano 4B exhibit recoverable thinking-branch entry states. Gemma and Nemotron transfer cleanly to broad non-AIME prompts, and all three models transfer to a disjoint contest-style prompt-string branch audit. Causal localization identifies the state at the final current-turn prompt positions, and module-split interventions show that attention-pathway patches account for the effect much more strongly than MLP-only patches. Sign, random, wrong-position, non-tail, token-mask, suffix, and adversarial visible-prompt controls distinguish the intervention from arbitrary suffix sensitivity, visible instruction following, and opener-token copying. Generation diagnostics show that related branch-state interventions alter response length, self-correction, and final-answer style. These results define a reproducible internal audit target for public reasoning controls: a compact residual state that can be measured before generation and used to test whether a public switch entered its advertised branch.

## 1 Introduction

Open language models increasingly expose user-facing controls for reasoning behavior. These controls may appear as a thinking button, a chat-template option, or an instruction that asks the model to reason before answering. Such controls affect latency, cost, response length, answer style, and users' expectations about whether the model has entered a more deliberative mode. However, surface behavior is not a sufficient audit target. A visible `<think>` tag, a long answer, or a concise final response can be produced or suppressed without establishing which internal state the model entered.

We distinguish three objects that are often conflated: the visible switch text, the internal branch state induced by that switch, and downstream generated behavior. The present paper studies the second object. We ask whether public thinking controls induce a measurable prompt-boundary state that can be localized and causally manipulated before long-form generation begins. This target is narrower than reasoning quality or task correctness, but it is correspondingly more mechanistic and more reproducible.

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

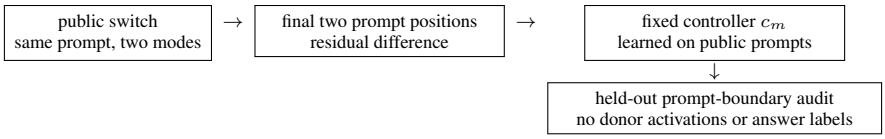


Figure 1: Main protocol. Public paired prompts train a fixed residual controller, and held-out prompt strings test whether the controller recovers the thinking-branch distribution. The held-out audit uses no answer labels, answer generations, or donor activations.

The audit uses paired mode runs for the same user prompt. From public training prompts, we learn a model-specific residual controller $c_m$ by averaging donor-minus-target differences at the final prompt positions. We then freeze $c_m$ and evaluate whether it moves held-out direct/off prompt-boundary states toward the paired thinking-mode next-token distribution. The held-out evaluation uses the target hidden state and a train-learned vector only; it withholds held-out donor activations, donor-target differences, generated answers, and correctness labels. We refer to this procedure as a *prompt-boundary branch-state audit*. If a model exposes a public thinking switch, the audit asks whether the switch produces a recoverable internal branch state.

This question is closely related to recent work on token-level reasoning triggers. Mid-Think shows that hybrid reasoning behavior can be influenced by small visible triggers such as `Okay`, `</think>`, and newline patterns (Yang et al., 2026). In contrast, the visible switch is the object under audit here rather than the intervention. We test the residual state induced by the public switch directly, using causal interventions at the prompt boundary. The main result is that thinking-mode entry can be localized, read out, and steered as a compact prompt-tail state. This provides an internal audit primitive for public reasoning controls: a switch can be tested by whether it writes the expected branch state before generation unfolds.

**Contributions.** This paper makes five contributions.

- We introduce a prompt-boundary branch-state audit for public reasoning controls in open language models.
- We show that Qwen3.5-4B, Gemma 4 E2B-it, and Nemotron Nano 4B instantiate model-specific residual controllers for thinking-mode entry that transfer from 60 public paired prompts to held-out prompt-boundary audits.
- We localize the causal state to the final current-turn prompt positions and show that attention-pathway patches carry the effect much more strongly than MLP-only patches.
- We separate branch-state control from simpler explanations using sign, random, wrong-position, non-tail, token-mask, suffix, and adversarial visible-prompt controls.
- We report generation diagnostics showing that related branch-state interventions change response length, self-correction, and answer style, and we release row-level artifacts and verification scripts for the headline claims.

Table 1: Audit overview. Each row gives one component of the evidence chain.

| Audit question | Main evidence | Result signal |
|---|---|---|
| Does a public switch write a recoverable thinking-entry state? | Fixed residual-tail controllers transfer from public paired prompts to held-out prompt-boundary audits. | Effective control and JS movement toward the paired thinking branch. |
| Where is the decisive state? | Width-2 tail patches flip branches, while matched non-tail and prefix controls are null. | Tail-position locality at the current-turn prompt boundary. |
| Which pathway carries the state? | Attention-only and MLP-only module-split patches. | Attention-pathway patches dominate MLP-only patches. |
| Is the effect a surface-level artifact? | Same-length suffix controls, token-mask audits, sign/random controls, and adversarial visible-prompt tests. | Structured residual control survives the surface controls. |
| Does branch-state steering affect generated behavior? | Public generation diagnostics. | Length, self-correction, and answer style move with the branch. |

We distinguish same-checkpoint public switches from paired-checkpoint comparisons. Qwen, Gemma, and Nemotron are same-checkpoint cases. OLMo uses separate Think and Instruct checkpoints to test whether the audit identifies an analogous branch state under a different entry path.

## 2 Setup

**Modes and models.** The primary same-checkpoint cohort contains three open model families with public thinking or reasoning controls: Qwen3.5-4B (Qwen Team, 2026), Gemma 4 E2B-it (Google DeepMind, 2026), and Llama-3.1 Nemotron Nano 4B v1.1 (NVIDIA, 2025). OLMo 3 7B (OLMo Team, 2026) is analyzed separately as a Think/Instruct checkpoint-pair comparison. Earlier Phi and Liquid experiments provide additional localization evidence in the appendix and supplement. Qwen exposes `enable_thinking`; under the chat template, the assistant suffix differs as follows:

| | |
|---|---|
| think | `<\|im_start\|>assistant\n<think>\n` |
| no-think | `<\|im_start\|>assistant\n<think>\n\n</think>\n\n` |

The shared prefix is intentional: in this template, no-think enters an empty think block and then closes it before the answer gap.

Throughout the paper, intervention labels are written as *donor mode into target mode*. For example, "think donor into no-think target" means that activations are copied from a think-mode run and inserted into the corresponding no-think run before reading out target-model logits. This convention avoids ambiguous shorthand used in some raw artifact filenames.

**Patch metric.** Let $p_D$ and $p_T$ be the donor and target final next-token distributions for the same user prompt under two modes, and let $p_P$ be the distribution after patching target residual state with donor information. We report donor win when

$$\mathrm{JS}(p_P, p_D) < \mathrm{JS}(p_P, p_T),$$

where JS is Jensen-Shannon divergence. We also report margin retention for low-rank patches, defined as the donor-vs-target JS margin recovered relative to the exact full tail patch. Confidence intervals are prompt-bootstrap intervals over stored row-level artifacts. They quantify variation over prompts for the fixed checkpoints, prompts, and intervention grid, not uncertainty over model training. Exact rates such as 60/60 or 0/60 are finite-suite audit results and should not be read as population guarantees.

**Mode-entry controllers.** We define the mode-entry operation operationally. The positive operation is "enter thinking mode." A model-specific residual controller $c_m$ realizes that operation in model $m$'s residual stream:

$$h_{\mathrm{tail}} \leftarrow h_{\mathrm{tail}} + \alpha c_m.$$

For the fixed prompt-boundary audit, $c_m$ is a single width-2 residual-tail vector learned from paired public prompts. Controllers are model-specific: Qwen, Gemma, Nemotron, and OLMo have different hidden dimensions and residual bases. The held-out test is falsifiable: after fitting $c_m$, the controller must steer unseen direct/off states toward thinking distributions, while the opposite-sign controller and matched random directions should fail under the same metric.

**Tail residual patches.** For a layer boundary $\ell$ and width $w$, a tail patch replaces the residual states at the final $w$ prompt positions in the target run with donor states. Matched non-tail controls patch the same number of positions away from the current-turn assistant suffix. Module-split patches separately replace attention or MLP pathway contributions at the decisive layer windows.

**Low-rank residual code.** For each prompt $i$, direction $d$, and decisive layer $\ell_d$, let

$$\Delta_i = \mathrm{vec}(h^D_{i,\ell_d,\mathrm{tail}} - h^T_{i,\ell_d,\mathrm{tail}}).$$

We fit an uncentered SVD basis to $\{\Delta_i\}$ on training prompts, then patch only the projection of each held-out prompt's exact donor-target difference into the learned subspace. This is a subspace-sufficiency test: at evaluation time, the held-out exact difference supplies the coordinates inside a basis learned only from training prompts. It therefore tests whether the causal mode information lies in a shared low-dimensional subspace, not whether a single fixed steering vector fully determines the branch.

**Fixed residual steering vector.** To test deployability more directly, we also learn a single vector per model and direction:

$$\bar{\Delta}_d = \frac{1}{|S_{\text{train}}|} \sum_{i \in S_{\text{train}}} \text{vec}(h^D_{i,\ell_d,\text{tail}} - h^T_{i,\ell_d,\text{tail}}).$$

At held-out evaluation time, we add $\alpha\bar{\Delta}_d$ to the target tail state, with $\alpha$ selected on training prompts only. This intervention uses the target hidden state and a fixed train-learned vector. It does not use held-out donor activations or held-out donor-target differences. For cross-family audit rows, we also report an "effective" rate: a prompt counts as effectively controlled only if the patched distribution has JS margin greater than 0.1 and JS to donor less than 0.6. This stricter metric requires meaningful movement toward the donor distribution rather than an infinitesimal donor win near maximum JS.

# 3 Visible switches induce transient branch states

The first Qwen experiments test whether the public switch is serialized in the chat template and whether the resulting transition states are distinguishable from arbitrary suffix noise. On 60 prompts, the no-think suffix has a deterministic transient path: immediately before the explicit close, the model predicts `</think>` on 60/60 prompts; immediately after the close, it predicts the blank separator on 60/60 prompts. The full no-think suffix then moves to direct-answer openers such as `To`, `Let`, and `Here`, while think mode moves to thinking-preface openers such as `Thinking` and `Here`.

Matched controls distinguish the serialized branch scaffold from generic suffix sensitivity. On a 180-prompt Qwen suite spanning arithmetic, algebra, calendar, coding, dynamic-programming, language, logic, planning, probability, and counting prompts, a real `</think>` outside the complete empty-block scaffold is close to no-think but not identical (mean JS to no-think 0.0534), fake close tags and moved-close controls do not reproduce canonical behavior, and same-length `Okay`, `Think`, and whitespace controls remain far from canonical think (mean JS to think 0.6519, 0.6767, and 0.6618). The effect is tied to a structured branch state induced by the actual serialized branch scaffold, rather than merely to changed final tokens.

Three further audits sharpen this interpretation. First, the Qwen fixed-controller public-prompt suite includes calendar, coding, dynamic-programming, language, logic-ordering, and planning prompts in addition to math. On those non-math and algorithmic/language categories, the fixed prompt-boundary controller reaches donor win 102/108 for no-think into think and 94/108 for think into no-think; the matched math/counting/probability categories reach 72/72 in both directions. Second, masked-JS audits test whether control is merely copied opener tokens. Table 2 shows that the train-learned branch controller keeps donor win 10/10 in both directions after renormalizing away special/whitespace tokens, after additionally masking canonical thinking/direct opener tokens, and after masking a broader hand-built branch-marker vocabulary. A visible verbose-vs-concise style vector, trained from no-think prompts with explicit style instructions, is weaker and collapses to chance after opener or branch-marker masking. A stricter learned-token mask ranks vocabulary items by train-split branch-distribution log-probability difference, then masks the top ranks on held-out prompts. For the fixed no-think-into-think controller, donor win and JS margin are 9/10 and .542 after masking the top 10 learned branch tokens, 8/10 and .293 after masking the top 100, and 9/10 and .411 after masking the top 250. Three norm-matched random controls never exceed 3/10 donor win and have negative margins. Branch-indicative tokens contribute to the JS gap, but the residual controller remains causal after broad branch-token masking. Third, an adversarial visible-prompt audit asks whether a residual branch-state classifier can track the real template branch when the user text says the opposite. Across 10 held-out prompts, seven prompt variants, and both modes, the residual classifier is correct on 140/140 cases, while a visible-instruction heuristic is correct on 60/140 and a first-token heuristic on 107/140. Ordinary verbosity/style directions, visible user requests, and opener tokens are therefore insufficient to account for the branch-state controller.

# 4 Tail-local causality and paired-checkpoint comparisons

Table 3 summarizes causal localization results for two same-checkpoint public switches and the secondary paired-checkpoint comparison. In Qwen, width-2 tail patches flip the public template branch from layer 0 or very early, while matched non-tail controls are null. Nemotron, whose

Table 2: Qwen masked-logit audit. Entries are held-out donor win and JS margin after renormalizing away increasingly broad token sets. The branch controller survives masking; a visible verbosity vector does not.

| Direction | Method | Unmasked | Mask openers | Mask branch markers |
|---|---|---|---|---|
| think into no-think | branch vector | 10/10, .571 | 10/10, .455 | 10/10, .455 |
| think into no-think | verbosity vector | 7/10, .037 | 5/10, .044 | 5/10, .044 |
| no-think into think | branch vector | 10/10, .064 | 10/10, .108 | 10/10, .108 |
| no-think into think | verbosity vector | 6/10, -.001 | 5/10, -.000 | 5/10, -.000 |

Table 3: Tail-local residual patching. Direction names are donor into target. Qwen and Nemotron are same-checkpoint public-switch localization rows. OLMo is a paired-checkpoint comparison.

| Model | Donor into target | Tail width | First perfect layer | Non-tail control |
|---|---|---|---|---|
| *Same-checkpoint public switch* | | | | |
| Qwen3.5-4B | think into no-think | 2 | 3 | 0.000 best donor win |
| Qwen3.5-4B | no-think into think | 2 | 0 | 0.000 best donor win |
| Nemotron Nano 4B | think into off | 2 | 12 | 0.000 best donor win |
| Nemotron Nano 4B | off into think | 2 | 16 | 0.000 best donor win |
| *Paired-checkpoint comparison* | | | | |
| OLMo 3 7B | think into instruct | 2 | 0 | 0.000 best donor win |
| OLMo 3 7B | instruct into think | 2 | 4 | 0.000 best donor win |

switch is a same-checkpoint system-prompt instruction, shows the same tail-local structure at later layer boundaries. OLMo is shown in a separate row group because it compares separately trained Think/Instruct checkpoints; its local branch state shows that the same audit can identify analogous states under checkpoint-level entry paths. Phi and Liquid boundary checks for other entry mechanisms are kept in the appendix and supplement. Prompt-bootstrap intervals and JS-to-donor/target distances for these rows are summarized in Appendix B.

Module-split patching narrows the pathway. In Qwen3.5-4B, attention-only width-2 patches reach donor win 1.000 by layer 16 in both directions, while MLP-only width-2 patches reach 0.000 for think donor into no-think target and at most 0.167 for the reverse. Nemotron repeats the same same-checkpoint pattern: attention-only width-2 patches reach donor win 1.000 in both directions, while MLP-only width-2 patches remain at 0.000. OLMo shows the same qualitative attention-pathway result across checkpoints. Single-head knockouts did not reveal a brittle one-head switch, so the localized state appears distributed across attention writes while remaining compact in residual space.

## 5 How the branch state is written and read

The causal account decomposes the intervention into writing, storage, and readout. The public switch writes a residual state into the final current-turn prompt positions: exact width-2 tail patches flip Qwen from the embedding boundary or very early and Nemotron by layers 12–16, while matched non-tail and prefix patches are null. The broad+GPQA Nemotron run sharpens the position claim: the fixed thinking-entry vector is 42/42 effective at the final two positions and 0/42 at offsets 8 and 16. The same vector is readable at several tested residual depths. Readout is primarily attention-pathway rather than MLP-pathway: Qwen and Nemotron attention-only width-2 patches reach donor win 1.000, while MLP-only patches are null or near-null. Head evidence is distributed across attention writes; direct logit attribution provides the vocabulary-level check, with Nemotron's off-to-think vector promoting `<th` by 17.9 logits and the reverse suppressing it by -8.95.

## 6 Broad prompt-boundary robustness

The main non-AIME audit trains controllers on 60 public GSM8K/MATH-500 paired prompts and evaluates on general instructions, coding, factual QA, multiturn, non-English, and GPQA-Diamond prompts. This is the primary out-of-domain robustness check. Thinking-entry transfer is clean for Gemma broad prompts (30/30 effective, JS-to-donor 0.0204, random 0/90) and Nemotron broad prompts (30/30 effective, JS-to-donor 0.0001, random 0/90); the reduced Nemotron broad+GPQA

Table 4: Main fixed-controller selection protocol. The grid and rule are fixed before held-out contest-style prompt evaluation. Effective control requires JS margin $> 0.1$ and JS-to-donor $< 0.6$.

| Model | Direction | Width grid | Fixed layer | Alpha grid / selected |
|---|---|---|---|---|
| Qwen3.5-4B | think into no-think | {2} | 15 | {.25,.5,1,2,4,8} / 1 |
| Qwen3.5-4B | no-think into think | {2} | 19 | {.25,.5,1,2,4,8} / 2 |
| Gemma 4 E2B-it | think into off | {2} | 35 | {.25,.5,1,2,4,8} / 2 |
| Gemma 4 E2B-it | off into think | {2} | 35 | {.25,.5,1,2,4,8} / 1 |
| Nemotron Nano 4B | think into off | {2} | 28 | {.25,.5,1,2,4,8} / 1 |
| Nemotron Nano 4B | off into think | {2} | 28 | {.25,.5,1,2,4,8} / 2 |

Table 5: Held-out contest-style branch audit for model-specific same-checkpoint controllers. Each controller is trained on 60 public paired mode runs and evaluated on all 60 AIME 2025+2026 prompt strings only as prompt-boundary branch audits. The fixed-vector intervention uses no AIME answers, correctness labels, donor activations, or donor-target differences. Random controls use three norm-matched seeds per model.

| Model | Entry path | Think eff. | Think JS | Opp. eff. | Random eff. |
|---|---|---|---|---|---|
| Qwen3.5-4B | template suffix | 60/60 | 0.1103 | 0/60 | 19/180 |
| Gemma 4 E2B-it | template switch | 60/60 | 0.0002 | 0/60 | 0/180 |
| Nemotron Nano 4B | system prompt | 60/60 | 0.0000 | 0/60 | 0/180 |

run reaches 42/42 effective with wrong-position controls at 0/42 and random controls at max 0/16. Qwen contributes the strongest exact-patch localization, suffix controls, token-mask controls, and adversarial branch-state audit; the per-family transfer pattern is summarized with the other claim boundaries in Section 11.

# 7 Held-out contest-style branch audit

We use AIME 2025+2026 prompts as a disjoint contest-style prompt set, but only for prompt-boundary branch-distribution auditing rather than answer generation or grading. Table 4 gives the selection protocol. Width is fixed at 2 from the prior tail-localization experiments; the layer is fixed per model and direction before held-out evaluation; the only scalar selected by the controller run is $\alpha$, chosen on the 60 public training prompts by lexicographically maximizing effective control, donor win, JS margin, and then minimizing JS-to-donor. No AIME prompts, answers, donor activations, or donor-target differences enter this selection. Table 5 reports the strict held-out test for the three same-checkpoint public switches: each model learns $c_m$ as a mean width-2 tail residual difference on public paired training prompts, selects $\alpha$ on those training prompts, and evaluates all 60 AIME 2025+2026 prompt boundaries by adding $\alpha c_m$ to the direct/off target tail state.

This stress test asks whether the same semantic operation, "thinking on," survives a disjoint prompt source. The controllers require only 60 paired public prompts per model and succeed on 180/180 contest-style branch-control cases. The controllers are fixed before this held-out evaluation and are rejected by sign and random-vector controls. Opposite-sign interventions have effective rate 0/180. Random norm-matched controls use three seeds per model; they are much weaker at 19/540, and all 19 effective cases come from the noisier Qwen row.

**Secondary OLMo checkpoint-pair result.** OLMo 3 7B compares separately trained Think and Instruct checkpoints. The same controller protocol nevertheless finds a similar local branch state: OLMo's Think/Instruct pair reaches 60/60 held-out prompt-boundary controls for thinking-branch induction with mean JS-to-donor 0.0207, and 60/60 for the reverse direct-answer branch with mean JS-to-donor 0.2873. This paired-checkpoint result shows that the audit can identify analogous branch states even when the entry mechanism is checkpoint-level rather than an inference-time public switch.

The selection-bias audit is simple: width is fixed at 2, layers are selected before held-out evaluation, $\alpha$ is selected only on the 60 public paired training prompts, the effective-control threshold is global, and AIME prompt strings are disjoint from training and withheld from donor activations, answers, and donor-target differences.

Table 6: Cross-validated low-rank residual subspace results. CIs are prompt-bootstrap intervals. Random subspaces of matched rank achieve donor win 0.000 in all rows.

| Model | Donor into target | Rank | Donor win | Margin retention |
|---|---|---|---|---|
| Qwen3.5-4B | think into no-think | 2 | 1.000 [1.000, 1.000] | 0.966 [0.906, 1.046] |
| Qwen3.5-4B | no-think into think | 2 | 1.000 [1.000, 1.000] | 0.910 [0.869, 0.948] |
| OLMo 3 7B | think into instruct | 2 | 1.000 [1.000, 1.000] | 0.862 [0.811, 0.908] |
| OLMo 3 7B | instruct into think | 1 | 1.000 [1.000, 1.000] | 1.021 [0.963, 1.074] |

# 8 Low-rank support

The fixed-vector audit above is the main deployable prompt-boundary test. A separate low-rank subspace analysis asks a more mechanistic question: whether the exact held-out mode difference lies in a shared low-dimensional causal subspace. Table 6 gives that result. We run 3-fold category-balanced cross-validation over a 60-prompt suite; each fold trains on 40 prompts and evaluates on 20 held-out prompts. The learned spectrum is concentrated in Qwen, where the top two components explain 0.874 and 0.854 of donor-target tail-difference energy in the two directions. OLMo has a similarly compressed paired-checkpoint subspace. The subspace interventions transfer causally; unlike the fixed-vector audit, they use the held-out prompt's exact donor-target difference to choose coordinates inside the train-learned subspace, so they support the low-rank mechanism rather than fixed-vector deployability.

A larger Qwen-only 180-prompt subspace suite clarifies what the code represents. Full width-2 patches have donor win 1.000 with JS to donor near zero. Rank-2 SVD patches also reach donor win 1.000 in both directions; matched random rank-2 subspaces remain at donor win 0.000. Wrong-prompt exact-difference patches transfer strongly, reaching donor win 0.956 and 0.994 in the two directions. The branch code is therefore shared rather than prompt-keyed, consistent with the fixed-vector result above.

Logit attribution of the rank-2 Qwen code is branch-vocabulary-shaped. The first component in the think-donor direction promotes tokens such as `Thinking`, `thinking`, and user-analysis tokens, while suppressing direct-answer openers such as `To`, ` To`, and `###`. This ties the residual subspace to the visible response program rather than leaving it as an abstract donor-win vector.

# 9 Generated behavior diagnostics

Prompt-boundary control tests the entry state before sampling. We therefore treat generation as a diagnostic rather than the main audit target. On public Qwen prompts, the serialized no-think transient states produce shorter, more direct, less self-correcting responses than the think suffix, and generated-prefix residual control shifts response style after sampled tokens begin to accumulate. These results connect the branch-state audit to visible completions, while the detailed tables are moved to Appendix D.

# 10 Related work

Hybrid-reasoning work studies when models should allocate reasoning budget and how visible triggers steer that allocation (Jiang et al., 2025; Yang et al., 2026). Mid-Think is the closest visible-trigger comparison: it shows that token-level triggers can change reasoning style without training, while our audit asks whether a public switch has produced a recoverable residual branch state after the visible trigger is already fixed. Causal rationale and bypass studies ask how much later computation depends on explicit reasoning traces (Rahim and Rahim, 2025; Sathyanarayanan et al., 2026); our target is the prompt-boundary state that enters the branch before long generated traces accumulate. Methodologically, we build on activation patching, activation addition, and layerwise readout (Geva et al., 2023; Belrose et al., 2023; Zhang and Nanda, 2024), while adding a held-out fixed-controller audit with sign, random, non-tail, masked-logit, suffix, and visible-prompt controls.

## 11 Limitations

The audit target is prompt-boundary thinking-mode entry, not answer correctness or reasoning faithfulness. The AIME 2025+2026 rows use contest-style prompt strings only as branch-distribution inputs: they do not generate answers, grade solutions, or use correctness labels. Accuracy appears only in generated-behavior diagnostics, where it checks complete responses and helps separate style movement from task performance.

The controllers are model-specific and direction-specific. Qwen, Gemma, and Nemotron are same-checkpoint public-switch cases, while OLMo is a paired Think/Instruct checkpoint comparison. Qwen and Nemotron support fixed direct-answer erasure in addition to thinking-entry induction; Gemma's reverse direct-answer state is exact-patchable with full tail residuals but is not captured reliably by one fixed mean vector. The strongest fixed-vector claim is therefore thinking-entry control, with reverse directions treated as calibration cases.

Broad non-AIME transfer varies by family. Gemma and Nemotron transfer cleanly on the broad prompt-boundary audit, including broad instruction/coding/multiturn/non-English prompts and Nemotron GPQA-Diamond prompts. Qwen has strong exact-patch localization, suffix/token controls, adversarial branch-state classification, and contest-style prompt-boundary transfer, but its single fixed mean vector is more prompt-distribution sensitive on the 30-prompt broad split (13/30 effective for thinking induction; random controls 2/90).

Generated-prefix control is a harder closed-loop setting than prompt-boundary control. The reported rollout diagnostics show movement in length, self-correction, and answer style, and one useful Qwen public-task controller, but accuracy improvements are not robust across directions and models. Sustained residual control after many generated tokens should be treated as a separate deployment problem.

The circuit localization is partial. We localize tail position, show low-rank residual structure, and show attention-pathway sufficiency relative to MLP-only patches. We do not identify a single-head circuit, a universal architecture-independent residual basis, or a benchmark-solving policy. Visible tokens also remain part of the entry path in several models: the claim is that their effect is mediated by recoverable residual branch states, not that the visible switch text is irrelevant. Proprietary systems may implement public reasoning controls differently.

## 12 Broader impacts

This work gives a concrete audit target for hybrid-reasoning systems: a compact residual controller is easier to test than a vague claim that thinking is on. The main dual-use risk is superficial manipulation of thinking-style behavior, so we pair the result with controls, held-out tests, and rollout diagnostics that distinguish branch entry from answer quality.

## 13 Conclusion

In the same-checkpoint public switches studied here, thinking-mode entry writes a reusable prompt-tail branch state. In Qwen, Gemma, and Nemotron, model-specific linear residual controllers enter the thinking branch from direct/off states; Gemma and Nemotron transfer cleanly on broad non-AIME prompt-boundary audits, and all three transfer on the disjoint contest-style branch audit without held-out donor activations. The controls rule out visible-instruction, opener-copying, random-vector, and generic suffix explanations, while localization results identify a tail-local attention-pathway state rather than a diffuse response-length artifact. Public reasoning switches can therefore be audited as internal branch-state transitions, using compact residual controllers measured before long generation begins.

## References

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In *EMNLP*, 2023.

Lingjie Jiang, Xun Wu, Shaohan Huang, Qingxiu Dong, Zewen Chi, Li Dong, Xingxing Zhang, Tengchao Lv, Lei Cui, and Furu Wei. Think only when you need with large hybrid-reasoning models. *arXiv preprint arXiv:2505.14631*, 2025.

Google DeepMind. Gemma thinking capabilities documentation and Gemma 4 E2B-it model card. `https://ai.google.dev/gemma/docs/capabilities/thinking`; `https://huggingface.co/google/gemma-4-E2B-it`, 2026.

Liquid AI. LFM2.5 1.2B Thinking model card. `https://huggingface.co/LiquidAI/LFM2.5-1.2B-Thinking`, 2026.

Microsoft. Phi-4-mini-reasoning model card. `https://huggingface.co/microsoft/Phi-4-mini-reasoning`, 2025.

NVIDIA. Llama-3.1-Nemotron-Nano-4B-v1.1 model card. `https://huggingface.co/nvidia/Llama-3.1-Nemotron-Nano-4B-v1.1`, 2025.

OLMo Team. OLMo 3 7B Think model card. `https://huggingface.co/allenai/Olmo-3-7B-Think`, 2026.

Qwen Team. Qwen3.5-4B model card. `https://huggingface.co/Qwen/Qwen3.5-4B`, 2026.

Noor Rahim and Ali Abdul Rahim. Dormant reasoning circuits in RL-trained language models. In *FoRLM*, OpenReview, 2025. `https://openreview.net/forum?id=mTjGBrkdtz`.

Anish Sathyanarayanan, Aditya Nagarsekar, and Aarush Rathore. Bypassing the rationale: Causal auditing of implicit reasoning in language models. *arXiv preprint arXiv:2602.03994*, 2026.

Wang Yang, Debargha Ganguly, Xinpeng Li, Chaoda Song, Shouren Wang, Vikash Singh, Vipin Chaudhary, and Xiaotian Han. Mid-Think: Training-free intermediate-budget reasoning via token-level triggers. *arXiv preprint arXiv:2601.07036*, 2026.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *ICLR*, 2024.

# A Artifact map

The row-level artifacts are stored under `artifacts/`. Each artifact directory contains run metadata, command logs, aggregate JSON files, and row-level JSON files. For double-blind review, the submitted supplement should be built with `scripts/prepare_neurips_supplement.sh`; this script copies reproduction scripts, selected row-level and aggregate artifacts, prompt splits, and sanitized run metadata while excluding paper source, written appendices, raw upload logs, and machine-specific shell logs.

The supplement also includes `scripts/verify_claim_artifacts.py`, which checks that the artifact directories and primary files in the artifact map below are present, and `scripts/verify_headline_numbers.py`, which checks the paper's rounded headline values against the stored JSON artifacts.

| Claim | Artifact directory | Primary aggregate files |
|---|---|---|
| Qwen tail localization | qwen35-paper-interp-20260422T075824Z | width_patch_aggregate.json |
| Qwen token controls and 180 prompts | qwen35-neurips-hardening-20260425T172250Z | token_control_aggregate.json, crossval_aggregate.json |
| Qwen fixed steering vector | qwen35-fixed-controller-20260427T152833Z | fixed_prompt_controller_aggregate.json, fixed_prompt_controller_bootstrap_summary.json |
| Qwen generated-prefix rollout diagnostic | qwen35-multilayer-controller-benchmark-20260427T233149Z | generation_aggregate.json, generation_rows.json |
| Gated public-task steering policy | gated-controller-policy-20260505T050337Z | gated_policy_aggregate.json, gated_policy_aggregate_by_model.json, gated_policy_rows.json, crossvalidated_gated_policy_rows.json, gated_policy_paired_deltas.json |
| Cross-family fixed vectors | cross-family-fixed-controller-20260427T195227Z | fixed_controller_aggregate.json, fixed_controller_rows.json |
| Contest-style prompt-string branch-state audit | aime-fixed-branch-controller-20260429T194405Z | aime_controller_aggregate.json, aime_controller_rows.json, controller_vectors.json |
| Gemma/Nemotron fixed thinking-branch switch | gemma-nemotron-fixed-switch-20260505T070908Z | aime_controller_aggregate.json, aime_controller_rows.json, controller_selection.json, controller_vectors.json |
| Broad branch-state claim synthesis | branch-state-broad-claim-20260506T034229Z | branch_state_broad_claim_summary.json, SUMMARY.md |
| Branch-entry robustness audits | branch-entry-paper-audits-20260506T210000Z | paper_audit_summary.json, SUMMARY.md, svd_top_token_proxy.json |
| Qwen masked-JS/style control audit | qwen35-masked-js-style-control-20260506T215634Z | masked_js_style_aggregate.json, masked_js_style_rows.json, mask_metadata.json |
| Qwen learned branch-token mask audit | qwen35-learned-token-mask-20260507T040000Z | learned_mask_aggregate.json, learned_mask_rows.json, learned_branch_tokens_top250.json |
| Qwen adversarial branch audit | qwen35-adversarial-branch-audit-20260507T000000Z | classifier_aggregate.json, patch_aggregate.json, SUMMARY.md |
| Nemotron broad+GPQA robustness | prompt-boundary-robustness-nemotron-reduced-20260507T061500Z | fixed_aggregate.json, random_direction_summary_aggregate.json, subset_aggregate.json, wrong_position_aggregate.json |
| Historical direct-erasure adapter diagnostics | shared-latent-step-direct-erasure-20260430T040633Z | shared_latent_aggregate.json, shared_latent_rows.json, latent_adapters.json |
| Historical Nemotron adapter stress test | shared-latent-nemotron4b-direct-erasure-step-20260505T040225Z | shared_latent_aggregate.json, shared_latent_rows.json, latent_adapters.json |
| Closed-loop direct-erasure diagnostic | closed-loop-shared-direct-erasure-reduced-20260505T041600Z | closed_loop_step_aggregate.json, closed_loop_prompt_aggregate.json, closed_loop_prompt_rows.json |
| Historical Phi kernel induction diagnostic | shared-latent-step-thinking-phi-kernel-layer28-20260430T052209Z | shared_latent_aggregate.json, shared_latent_rows.json, latent_adapters.json |
| OLMo localization | olmo3-localization-20260423T160830Z | block_patch_aggregate.json, module_patch_aggregate.json |
| Nemotron localization | nemotron-localization-20260507T000000Z | patch_aggregate.json, module_patch_aggregate.json, first_perfect_layers.json |
| Liquid localization | liquid-causal-patch-20260426T044355Z, liquid-causal-layer-scan-20260430T051648Z | patch_aggregate.json, canonical_aggregate.json |
| Phi localization | phi-causal-patch-20260423T231120Z | patch_aggregate.json |
| Low-rank cross-validation | residual-subspace-crossval-20260424T205031Z | crossval_aggregate.json, spectrum_aggregate.json |
| Public behavior diagnostics | qwen35-public-benchmark-behavior-20260424T104510Z | aggregate_by_variant.json |
| Sustained patch generation | qwen35-sustained-patch-generation-20260425T193507Z | generation_aggregate.json, sustained_step_aggregate.json |

# B Localization effect sizes

Table 7 expands the headline localization rows from Table 3. For Qwen, JS columns are point estimates from the stored aggregate. For Nemotron, OLMo, Liquid, and Phi, the corresponding aggregate files include prompt-bootstrap intervals. Phi reports the main 8-prompt set; the alternate set reaches the same first perfect layers.

| Model | Donor into target | $n$ | Layer | Donor win | JS to donor | JS to target |
|---|---|---|---|---|---|---|
| Qwen3.5-4B | think into no-think | 60 | 3 | 1.000 [1.000, 1.000] | 0.0632 | 0.6283 |
| Qwen3.5-4B | no-think into think | 60 | 0 | 1.000 [1.000, 1.000] | 0.1113 | 0.6080 |
| Nemotron Nano 4B | think into off | 12 | 12 | 1.000 [1.000, 1.000] | 0.1222 | 0.6748 |
| Nemotron Nano 4B | off into think | 12 | 16 | 1.000 [1.000, 1.000] | 0.1404 | 0.6893 |
| OLMo 3 7B | think into instruct | 12 | 0 | 1.000 [1.000, 1.000] | 0.2552 | 0.6925 |
| OLMo 3 7B | instruct into think | 12 | 4 | 1.000 [1.000, 1.000] | 0.5022 | 0.6927 |
| Liquid LFM2.5 1.2B | think into instruct | 60 | 16 best | 0.000 [0.000, 0.000] | 0.6924 | 0.6037 |
| Liquid LFM2.5 1.2B | instruct into think | 60 | 12 | 1.000 [1.000, 1.000] | 0.1246 | 0.6931 |
| Phi-4 mini | think into instruct | 8 | 28 | 1.000 [1.000, 1.000] | 0.0110 | 0.6932 |
| Phi-4 mini | instruct into think | 8 | 16 | 1.000 [1.000, 1.000] | 0.2996 | 0.6921 |

Table 7: Effect sizes for the width-2 tail residual localization rows. The Liquid think-into-instruct row reports the best tested tail layer rather than a first perfect layer, since no tested layer produced donor wins. Matched non-tail controls have donor win 0.000 [0.000, 0.000] for all rows with bootstrap intervals in the artifacts.

# C Compute and existing assets

Experiments were run on an Apple M4 Mac Mini with 16GB unified memory using MLX and MLX-LM. The BF16 Qwen replication is narrower than the 4-bit cross-validation because of this memory limit. The sustained-generation suite is also smaller because it recomputes full donor and target traces at each generated step. The supplement's per-run `run_metadata.json` files record UTC start times, platform strings, model identifiers, layer and alpha grids, prompt counts, and generation caps for the included runs.

The model checkpoints are open Hugging Face assets: Qwen3.5-4B is Apache-2.0, OLMo 3 7B Think/Instruct is Apache-2.0 with Ai2 Responsible Use Guidelines, Gemma 4 E2B-it is documented by Google DeepMind with thinking-mode controls, LiquidAI LFM2.5 1.2B uses the LFM Open License v1.0, Phi-4-mini-reasoning is MIT, and Llama-3.1-Nemotron-Nano-4B-v1.1 uses the NVIDIA Open Model License with additional Llama 3.1 terms. The prompt-boundary branch audits use math-ai/aime25 and math-ai/aime26, which are Apache-2.0 licensed on their Hugging Face dataset pages. Public generation diagnostics are provided as behavior artifacts for the same intervention family.

# D Generated behavior diagnostics

Generated-prefix control tests how the intervention family behaves after sampled tokens change the future state distribution. In a Qwen rollout diagnostic, the controller is trained on 24 public prompts, evaluated on 40 disjoint GSM8K/MATH-500 prompts, added for the first generated decision and then for up to 64 generated-prefix decisions, and then allowed to continue freely. The selected no-think-to-think generated-prefix controller changes response style and has favorable held-out accuracy in this public setting while keeping truncation far below canonical think; the paired reverse row distinguishes style movement from answer-quality improvement.

| Condition | Acc. | Direct | Self-corr. | Tokens | Trunc. |
|---|---|---|---|---|---|
| canonical no-think | 0.725 | 0.925 | 0.075 | 530.5 | 0.150 |
| canonical think | 0.200 | 0.125 | 0.850 | 1012.8 | 0.950 |
| no-think $\rightarrow$ think generated-prefix controller | 0.800 | 0.000 | 0.000 | 516.1 | 0.100 |
| think $\rightarrow$ no-think generated-prefix controller | 0.075 | 1.000 | 0.225 | 1013.0 | 0.950 |

The serialized-suffix diagnostic asks whether downstream behavior moves after the prompt-boundary state enters the transient no-think branch. On a separate 24-prompt public subset, canonical Qwen think is much longer and more self-correcting but truncates on 75.0% of prompts; no-think transient states are shorter, less self-correcting, and match no-think-style direct-answer behavior. Accuracy is included to confirm complete responses.

| Variant | Think | Direct | Self-corr. | Acc. | Tokens | Trunc. |
|---|---|---|---|---|---|---|
| think suffix | 0.750 | 0.167 | 0.917 | 0.417 | 3585.9 | 0.750 |
| shared prefix | 0.208 | 0.708 | 0.500 | 0.500 | 2022.2 | 0.375 |
| no-think preclose | 0.000 | 0.875 | 0.167 | 0.708 | 984.4 | 0.125 |
| closed, no gap | 0.000 | 0.875 | 0.167 | 0.708 | 1028.0 | 0.125 |
| full no-think suffix | 0.000 | 0.875 | 0.250 | 0.708 | 1071.7 | 0.125 |

# NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction state the positive claim as a prompt-boundary branch-state audit for thinking-mode entry with model-specific residual-tail controllers. The limitations section scopes this claim by separating prompt-boundary branch control from rollout correctness, bidirectional erasure, and contest-math solving.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: The limitations section discusses model-family scope, paired-checkpoint versus same-checkpoint evidence, direction-specific controller asymmetries, Qwen's mixed broad fixed-vector transfer, generated-prefix diagnostics, and the distinction between prompt-boundary branch-state control and rollout correctness.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A].

   Justification: The paper is empirical and does not present theoretical theorems or proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: The setup defines the intervention metrics, patch widths, low-rank and fixed-vector fitting procedures, model families, prompt splits, and generated-prefix diagnostic setting; the artifact map points to scripts, row-level JSON files, aggregate JSON files, and run metadata for the reported results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: The experiments use open model checkpoints and public datasets where applicable. The anonymous supplement contains reproduction scripts, selected row-level and aggregate artifacts, prompt splits, and a README with fast verification commands and model-level reproduction requirements. Written appendices and the checklist are included in the main PDF rather than in the supplement.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The paper states the 3-fold 40/20 prompt cross-validation, fixed-vector train/eval splits, generated-prefix diagnostic setting, tail width 2, key layer boundaries, prompt counts, and generation caps. The supplement includes the corresponding experiment scripts, prompt splits, aggregate and row-level artifacts, and sanitized run metadata.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

   Justification: The main low-rank and fixed-vector tables report prompt-bootstrap confidence intervals where space permits, the setup defines the bootstrap procedure, and the text reports finite-suite caveats for exact 60/60 or 0/60 rates. Null opposite-sign, random, wrong-position, and non-tail controls are reported alongside the learned interventions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The appendix states that the reported runs used MLX/MLX-LM on an Apple M4 Mac Mini class machine with 16GB unified memory. The included run metadata records start timestamps, platform, model identifier, prompt counts, patch grids or generation caps where applicable, and the paper notes the memory-driven BF16 and generation-cap constraints.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

Justification: The work studies open model internals and public benchmarks, does not involve human subjects, and does not train or release a new high-risk model.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes].

Justification: The Broader Impacts section discusses the positive auditability benefit and the dual-use risk that better understanding of public thinking switches could improve both oversight and manipulation of thinking-style behavior.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A].

Justification: The paper does not release a new pretrained model, scraped dataset, or generative system. It releases analysis scripts and small artifacts derived from open checkpoints.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes].

    Justification: The appendix lists the model checkpoints and public branch-audit datasets used, with license notes for Qwen, OLMo, Gemma, LiquidAI, Phi, Nemotron, and math-ai AIME 2025/2026 where available from the source cards. Public generation diagnostics are documented as behavior artifacts for the same intervention family.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes].

    Justification: The new assets are experiment scripts and derived activation-analysis artifacts. They are organized by run directory with summaries, sanitized metadata, aggregate files, row-level files, prompt splits, and selected reasoning-mode adapter metadata.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing or human-subject experiments.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The paper does not involve human-subject research and therefore does not require IRB approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A].

    Justification: LLMs were not important, original, or non-standard components of the core method development in this research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.