

LLMs Decide Early and Explain Later: You Can Stop Them Early

Ayan Datta^{1*}, Zhixue Zhao^{2*}, Mounika Marreddy³, Bhuvanesh Verma³,
Radhika Mamidi¹, Alexander Mehler³

¹IIIT Hyderabad, ²University of Sheffield, ³Goethe University

advin4603@gmail.com, zhixue.zhao@sheffield.ac.uk, mmarredd@em.uni-frankfurt.de,
verma@em.uni-frankfurt.de, radhika.mamidi@iiit.ac.in, mehler@em.uni-frankfurt.de

Abstract

Large Language Models often achieve strong performance by generating long intermediate chain-of-thought (CoT) reasoning. However, it remains unclear when a model’s final answer is actually determined during generation. If the answer is already fixed at an intermediate stage, subsequent reasoning tokens may constitute post-decision explanation, increasing inference cost and latency without improving correctness. We study the evolution of predicted answers over reasoning steps using *forced answer completion*, which elicits a model’s intermediate prediction at partial reasoning prefixes. Across a range of models and datasets, we find that the final answer is fixed from the earliest stages of generation in most cases, with **only 32% of answers changing after CoT**. Moreover, once the final answer switch does occur, models continue to generate substantial reasoning, producing **760 additional tokens per query** on average and accounting for around 40% of the total reasoning. Motivated by these findings, we explore early-stopping strategies that halt generation once the answer has stabilized. We show that simple probe-based stopping can reduce reasoning token usage by **500 tokens per query** while incurring only a **2% drop in accuracy**. Together, our results indicate that a large portion of chain-of-thought generation functions as **post hoc explanation** and can be reduced with minimal impact on performance.

1 Introduction

Large language models (LLMs) trained for explicit reasoning have achieved strong performance across a wide range of reasoning tasks, including mathematical problem solving, multiple choice question answering, and tool assisted decision making (Wei et al., 2022; Plaat et al., 2025; Lewkowycz et al., 2022; Kojima et al., 2022), by generating intermediate reasoning steps before producing a final answer.

*Equal contribution

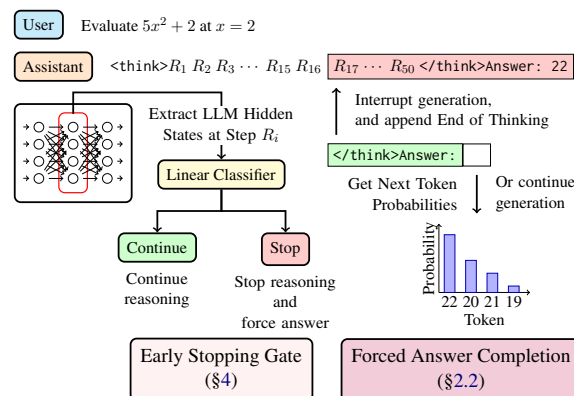


Figure 1: The model reasons in steps $R_1, \dots, R_n$ before answering. **Left (Early Stopping Gate):** a linear probe over the step- R_i layer l hidden state scores whether further reasoning is needed and halts generation when confident. **Right (Forced Answer Completion):** we append the end-of-thinking token and an answer prefix, then read the answer from the next-token distribution. We find that late tokens ($R_{17} \dots R_{50}$) are largely post-decision and can be saved.

While these traces are typically treated as a monolithic process, the temporal relationship between individual reasoning steps and the model’s predicted answer remains poorly understood. Despite their empirical success, the role of these intermediate reasoning tokens in the actual decision-making process of LLMs is unclear.

Although longer reasoning is often associated with higher accuracy, it is unclear whether every generated token actually updates the model’s answer, or whether models keep producing coherent reasoning after their decision has effectively stabilized. Yet most analyses treat reasoning as a static object and ask whether the final explanation is faithful, rather than tracking *when* the answer is decided. As a result, there has been limited work investigating how a model’s prediction stabilizes over the course of the reasoning process.

To address this, we use *forced answer comple-*

tion, a method that intervenes on partial reasoning prefixes to elicit the model’s predicted answer at intermediate steps. By applying this procedure throughout a reasoning trace, we obtain an *answer trajectory* that records how the predicted answer changes over reasoning steps generated. This makes it possible to measure answer switches, transient answer flips, and the point at which the predicted answer reaches its final decision.

Using answer trajectories, we empirically characterize the temporal dynamics of predicted answers throughout the model’s reasoning process across multiple task settings. We find that predicted answers change in only **32%** of queries on Qwen3-4B, indicating that in the majority of cases the model’s answer is fixed from the earliest stages of generation, and we observe qualitatively similar behavior across other models as well. When answer changes do occur, they typically happen early in the reasoning trace. Despite this early stabilization, the model continues to generate substantial additional reasoning, producing on average **760 tokens after the final answer switch**, and up to **1000 tokens** in some cases. This reveals a systematic temporal gap between when a model’s decision is made and when its explanation is produced. Figure 1 illustrates our methodology through an example in which a reasoning model’s answer stabilizes early in its reasoning process, after which our early-stopping gate detects this stabilization, halts further reasoning, and forces the model to produce the final answer. We make our implementation publicly available.¹

2 Methodology

2.1 Problem Setup and Notation

We study tasks in which a language model is given an input context and produces a final answer accompanied by an explicit reasoning trace. The input context is denoted by C , which includes the conversation history, task prompt, and any additional information provided to the model. Let R denote the reasoning trace generated by the model. We represent R as a sequence $R = (R_1, R_2, \dots, R_n)$, where each R_i corresponds to a distinct unit of reasoning. In this work, we assume individual sentences as the basic units of reasoning for analysis purposes, consistent with prior work on natural language rationales and chain-of-thought explana-

tions (Wei et al., 2022; Lampinen et al., 2022). Recent reasoning-oriented models explicitly separate reasoning from the final answer. In particular, models inspired by Shao et al. (2024) employ special `<think>` tags to delimit reasoning content. Under this formulation, the final assistant output can be written as: `<think>R</think>A`, where A denotes the final answer. Note that **a decision differs from the model’s internal belief**. We define the decision to be the final output the model surfaces at a given moment, whereas a belief is an internal state that influences decisions. While recovering a model’s true belief is valuable, it is not our goal here, as we are concerned only with what the model surfaces and not with its hidden belief.

2.2 Forced Answer Completion

Inspired by Lanham et al. (2023), we analyze the evolution of model predictions over the course of a reasoning trace. Given an input context C and a reasoning trace $R = (R_1, R_2, \dots, R_n)$, we estimate the intermediate answer the model would predict after each step by explicitly intervening to halt further reasoning and elicit an immediate final answer or decision. Concretely, after generating the partial reasoning trace $(R_1, \dots, R_i)$, we append the end-of-thinking token `</think>` followed by an answer prefix P , yielding the prompt

$$C\text{<think>}R_1R_2\dots R_i\text{</think>}P$$

Sampling the model conditioned on this prompt produces an intermediate answer A_i . Repeating this procedure after each reasoning step yields an *answer trajectory* $T = (A_0, A_1, \dots, A_n)$, where A_0 denotes the answer generated without any reasoning steps and A_i denotes the answer generated after step R_i . The answer prefix for each task is listed in Section B in the Appendix.

2.3 Measuring Decision Changes

Given an Answer Trajectory $T = (A_0, A_1, A_2, \dots, A_n)$ induced by forced answer completion, we define a set of metrics that characterize the temporal behaviour of the model’s predicted answer over the reasoning steps $R = (R_1, R_2, \dots, R_n)$.

2.3.1 Final answer switch index.

We define the index of the last answer switch as the index after which the model’s intermediate answer does not change. We calculate this as:

$$t^* = \max \{ i \in \{0, \dots, n-1\} \mid A_i \neq A_{i+1} \}, \quad (1)$$

¹<https://github.com/advin4603/LLMs-Decide-Early-And-Explain-Later>

with the convention that $t^* = -1$ if no such index exists (i.e., $A_0 = A_1 = \dots = A_n$).

2.3.2 Answer Switches

We calculate the number of times the model switches its answer from one to another. We do so using the following:

$$\text{AnswerSwitches}(T) = \sum_{i=0}^{n-1} [A_i \neq A_{i+1}] \quad (2)$$

Where $[\cdot]$ denotes the Iverson bracket, which evaluates to 1 if the condition is true, else 0.

2.3.3 Transient Answer Flips

We also detect cases where the predicted answer briefly switches to an alternative, e.g., for one more step, and then switches back. We refer to these locally inconsistent trajectories *Transient Answer Flips (TAFs)*. That is, a transient answer flip is a short-lived deviation from a dominant answer, lasting a small number of reasoning steps (k), after which the model returns to the original answer. Given an Answer Trajectory $T = (A_0, A_1, A_2, \dots, A_n)$, we calculate the number of TAFs as follows:

$$\text{TAFs}(T) = \sum_{i=0}^{n-k-1} \left[\exists \ell \leq k \text{ s.t. } A_i = A_{i+\ell+1} \wedge \forall j \in \{1, \dots, \ell\}, A_i \neq A_{i+j} \right] \quad (3)$$

Where k is the max length of TAFs we want to detect. We set $k = 3$ (We motivate this choice in Appendix D). Because transient answer flips reflect local instability rather than genuine decision changes, we apply a causal hold-for- k smoothing that accepts an answer switch only once it persists for k consecutive steps (we use $k=3$), suppressing short-lived flips while preserving sustained changes. Throughout, we report each metric on both the raw and the denoised trajectory. We defer the formal definition to Appendix D.

2.4 Tokens After Final Answer Switch

Given an answer trajectory $T = (A_0, \dots, A_n)$, we measure the number of tokens generated after t^* . Let T_i denote the cumulative number of generated reasoning tokens up to step i , with $T_0 = 0$ and $T_n = T_{\text{total}}$. We define: $T_{\text{after}} = T_{\text{total}} - T_{t^*+1}$. This quantity captures the amount of generated tokens that would be saved by halting reasoning immediately after the final answer switch, without altering the model’s predicted answer.

2.5 Datasets

We evaluated a total of four tasks. For each task, we report per-example trajectories $(A_0, \dots, A_n)$ under forced completion and apply identical evaluation and monitoring procedures unless stated otherwise.

Multiple-choice Questions (MCQ) MCQ is commonly used for reasoning evaluation, where each example consists of a question and a small fixed set of candidate options, typically $\{A, B, C, D\}$. We use the MMLU-pro dataset from (Wang et al., 2024), sampling the correct answer choice and three other distractor choices. For these datasets, we treat the answer space as the finite set of provided options. At each reasoning step i , we record the option with the highest probability under forced answer completion and use this option as the per-step label A_i .

Numeric-answer We use the PRM800k dataset (a subset of MATH) from (Lightman et al., 2023), where correct responses are predominantly numeric and typically represented by a single token (e.g., an integer or short numeric constant). For these tasks, we record the highest-probability token under forced completion at each generation step, yielding an answer trajectory $A_0, \dots, A_n$.

Search-query We evaluate on a closed-book question-answering dataset from (Monteiro et al., 2024) augmented with an explicit search-tool interface. Each example consists of a question paired with a collection of background documents. During generation, the model is prompted to invoke a search tool by producing an appropriate search query as a tool argument. For analysis, we treat the generated search query itself as the answer, and all the documents in the dataset as the knowledge base to be retrieved from.

Tool-selection Tool selection is a fundamental challenge in agent-based systems. We evaluate the reasoning capability of LLMs using tasks in which the model is given a fixed set of tools and must select exactly one tool as part of its response (QuotientAI, 2025). At each reasoning step, we extract the tool identifier produced by the model when prompted to emit a tool call. The selected tool is recorded as the per-step label A_i , and the set of available tools defines the answer space for these datasets.

Together, these tasks span a broad spectrum of the verifiable problems that reasoning models are

trained and evaluated on. Such models are primarily optimized for structured, verifiable tasks like mathematics, coding, science, and logical reasoning rather than open-ended creative generation, so we deliberately exclude creative writing and other open-ended tasks where no well-defined answer trajectory exists. The chosen datasets also cover a range of difficulty, containing a mix of easy, medium, and hard questions that demand different amounts of reasoning. We further evaluate on frontier-level, substantially harder datasets in Appendix F.

2.6 Models

We evaluate two model families with distinct training methodologies. From the Qwen3 family, trained with a GRPO-style objective, we use the dense variants Qwen3-4B, Qwen3-8B, Qwen3-14B and the MoE Qwen3-30B-A3B (Yang et al., 2025). From the Gemma-4 family, we evaluate Gemma-4-12B-it (Team et al., 2026). We additionally evaluate GPT-OSS-20B (OpenAI, 2025), which uses a proprietary reasoning method, at high and medium reasoning effort, excluding low as it generates little reasoning. Together these span 4B–30B parameters. These models were run on an NVIDIA L40S with 48GB of VRAM.

3 Results

We report results on how the model’s predicted answer evolves across reasoning steps using forced answer completion, and how these dynamics can be used to reduce the number of generated reasoning tokens. We analyze the frequency and distribution of answer switches and transient answer flips across tasks and models, and measure how many reasoning tokens are generated after the final answer switch. Building on these observations, we evaluate early stopping strategies that halt generation at intermediate reasoning steps and quantify the resulting trade-off between token usage and final answer accuracy.

3.1 Properties of Answer Trajectories

We discuss observed properties of answer trajectories obtained via forced answer completion, focusing on how predicted answers change across reasoning steps and how these changes are distributed across models and tasks.

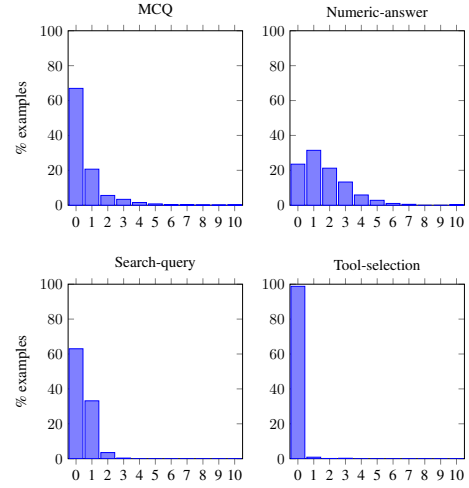


Figure 2: Distribution of answer switch counts across denoised answer trajectories for different task types using Qwen3-4B.

3.1.1 Transient Answer Flips

We first examine the occurrence of transient answer flips, defined as short-lived deviations in the predicted answer that revert within a small number of reasoning steps. Across all evaluated models and task settings, transient answer flips are common, indicating frequent, brief changes in the predicted answer that do not persist. Table 1 reports the average number of transient answer flips per example for each model across tasks. We observe that transient flips occur frequently, **≈ 3 TAFs per reasoning trace**, across all models for multiple-choice and numeric answers. We find a lower transient answer flip count for tool-use and search which is consistent with the lower answer switches.

3.1.2 Answer Trajectories with No Answer Switches

Next, we analyze the fraction of examples in which the predicted answer does not change at any reasoning step, i.e., the answer trajectory satisfies $A_0 = A_1 = \dots = A_n$ or $t^* = -1$. In such cases, the predicted answer under forced completion remains identical to the non-reasoning baseline throughout the entire reasoning trace. Figure 2 shows the distribution of the number of answer switches across examples. A substantial fraction of examples (**$\approx 50\%$ of examples in the MCQ dataset across all models**) exhibit zero answer switches, indicating that for these inputs the predicted answer is unchanged by the generated reasoning steps.

Table 1: Answer-trajectory Metrics computed across different datasets. For $\Pr(t^* = -1)$, we compute the fraction of examples with no answer switch on the raw traces and the denoised traces, respectively. T_{after} measures "tokens (percentage of reasoning tokens)" on samples where $t^* \neq -1$. We present detailed distributions in Appendix H

Dataset	Model	$\Pr(t^* = -1)$	AnswerSwitches Raw / Denoised	TAF(T)	$T_{\text{after}} (t^* \neq -1)$
		Raw% / Denoised%			Raw / Denoised
MCQ	Qwen3-4B	58.6 / 67.0	3.59 / 0.61	2.02	555.2 (35.7%) / 867.1 (43.1%)
	Qwen3-8B	50.5 / 59.2	4.30 / 0.80	2.29	663.0 (35.9%) / 1033.4 (43.4%)
	Qwen3-14B	55.4 / 65.3	3.11 / 0.64	1.64	609.9 (35.4%) / 783.5 (39.3%)
	Qwen3-30B-A3B	67.9 / 73.1	2.17 / 0.44	1.16	641.5 (40.1%) / 868.5 (45.5%)
	GPT-OSS-20B (H)	56.1 / 63.6	2.16 / 0.63	0.97	583.7 (47.0%) / 712.6 (46.3%)
	GPT-OSS-20B (M)	50.3 / 60.8	2.91 / 0.75	1.36	284.1 (35.7%) / 361.2 (31.2%)
	Gemma-4-12B-it	62.2 / 69.8	2.33 / 0.54	1.16	1843.4 (56.5%) / 2368.3 (60.3%)
Numeric-answer	Qwen3-4B	19.4 / 23.5	7.15 / 1.64	3.40	846.7 (42.6%) / 1192.9 (51.8%)
	Qwen3-8B	20.8 / 24.8	9.26 / 1.74	4.68	1280 (45.8%) / 1904.4 (59.9%)
	Qwen3-14B	18.2 / 27.3	8.77 / 1.80	4.43	1271.7 (50.5%) / 1717.3 (62.4%)
	Qwen3-30B-A3B	31.2 / 37.5	6.75 / 1.06	3.75	1203.8 (50.5%) / 1830.4 (70.2%)
	GPT-OSS-20B (H)	9.9 / 14.6	13.02 / 2.35	5.82	823.6 (39.5%) / 1437.5 (51.9%)
	GPT-OSS-20B (M)	8.0 / 13.3	13.83 / 2.72	5.77	307.0 (25.4%) / 466.8 (27.7%)
	Gemma-4-12B-it	18.8 / 25.4	7.92 / 1.43	3.80	2045.7 (51.1%) / 3256.9 (66.3%)
Search-query	Qwen3-4B	53.5 / 63.0	0.85 / 0.41	0.26	148.1 (64.7%) / 106.5 (44.7%)
	Qwen3-8B	60.1 / 74.5	0.87 / 0.28	0.38	145.5 (58.6%) / 121.6 (46.6%)
	Qwen3-14B	68.5 / 79.8	0.61 / 0.22	0.26	114.4 (54.4%) / 78.5 (36.3%)
	Qwen3-30B-A3B	60.1 / 80.2	0.91 / 0.22	0.40	95.8 (45.2%) / 82.2 (35.3%)
	GPT-OSS-20B (H)	74.7 / 97.3	0.55 / 0.03	0.29	36.8 (19.4%) / 63.4 (29.4%)
	GPT-OSS-20B (M)	88.4 / 99.6	0.17 / 0.00	0.07	9.1 (15.5%) / 12.4 (10.1%)
	Gemma-4-12B-it	81.6 / 99.2	0.29 / 0.01	0.10	12.7 (19.5%) / 8.3 (8.4%)
Tool-selection	Qwen3-4B	97.7 / 98.8	0.21 / 0.02	0.14	647.0 (41.3%) / 855.7 (56.6%)
	Qwen3-8B	97.4 / 98.6	0.11 / 0.02	0.06	338 (44.9%) / 465.3 (46.8%)
	Qwen3-14B	96.4 / 98.0	0.16 / 0.03	0.09	331.6 (44.6%) / 306.6 (38.5%)
	Qwen3-30B-A3B	99.0 / 100	0.02 / 0.00	0.01	37.0 (23.6%) / 37.0 (23.6%)
	GPT-OSS-20B (H)	87.3 / 96.1	0.79 / 0.08	0.44	576.9 (53.1%) / 677.8 (54.8%)
	GPT-OSS-20B (M)	93.0 / 97.1	0.31 / 0.05	0.17	169.5 (49.6%) / 147.2 (39.1%)
	Gemma-4-12B-it	75.1 / 87.6	1.69 / 0.19	0.97	366.7 (40.0%) / 495.0 (47.8%)

3.1.3 Limited Number of Answer Switches When Switches Occur

We consider examples in which the predicted answer changes at least once. Among these, the total number of answer switches is typically small relative to the length of the reasoning trace. Figure 2 shows the distribution of answer switches. The majority of such examples exhibit only a small number of switches, with long stretches of reasoning steps between successive changes. Denoising removes transient flips and concentrates the distribution on smaller switch counts (e.g., average answer switches drop from ≈ 8 to ≈ 2 on Numeric-answer), while preserving sustained changes. We report the per-dataset effect in Appendix E and show a worked example in Figure 3.

3.1.4 Tool-Selection Tasks Exhibit Near-Zero Answer Switches

For tool-selection datasets, from Table 1 we observe that the predicted answer is highly stable across reasoning steps as seen from ≈ 0 Answer Switches for the Tool Selection Dataset across all models. In the vast majority of examples, the model selects the same tool from the initial forced completion onward, resulting in almost no answer switches

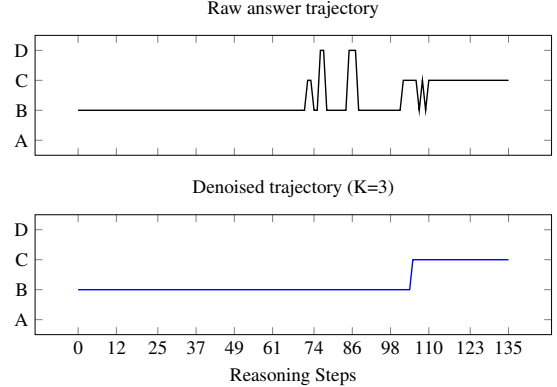


Figure 3: Denoising a Qwen3-4B MCQ trace.

throughout the reasoning trace.

3.2 Tokens After Final Answer Switch

We now analyze how many reasoning tokens are generated after the model’s final answer switch, as defined in Section 2.3.1. For each example, we measure the number of tokens generated after the last reasoning step at which the predicted answer changes.

Table 1 shows the average number of tokens generated after the final answer switch per example. We observe that a large fraction of the total reason-

ing tokens ($\approx 50\%$ of reasoning tokens across all dataset and models) are generated after the predicted answer has already reached its final value.

These tokens represent reasoning that does not alter the model’s predicted answer under forced completion. As a result, the measured quantity provides an upper bound on the number of tokens that could be avoided by an ideal early stopping strategy that halts generation immediately after the final answer switch.

While practical early stopping methods may not perfectly identify this point, the observed distributions indicate that a substantial portion of generated reasoning tokens occur after the final answer switch, leaving significant room for reducing token usage without changing the predicted answer.

4 Early Stopping and Performance Trade-Offs

We evaluate two early-stopping strategies for determining when to terminate a model’s reasoning process, i.e., when intermediate predictions are redundant to the final decision.

4.1 Early Stopping Gates

We introduce early-stopping gates that operate over partial reasoning prefixes and produce a scalar confidence score indicating whether generation can be terminated at a given step. These gates do not alter the model’s generation process; rather, they observe intermediate states and define stopping rules based on predefined criteria.

Formally, at each reasoning step $i \in \{0, \dots, n\}$, a gate observes information available up to that step and outputs a scalar score:

$$m_i = \mathcal{M}(C, R_1, \dots, R_i), \quad m_i \in [0, 1], \quad (4)$$

where $\mathcal{M}$ denotes the gate function. The score m_i is interpreted as the gate’s confidence that further reasoning is unnecessary.

Given a threshold $\tau \in [0, 1]$, the corresponding stopping rule halts generation at the earliest step

$$s = \min\{i \mid m_i \geq \tau\}. \quad (5)$$

Varying τ allows explicit control over the tradeoff between final answer accuracy and token savings.

We consider the two gating criteria, random gating and Probe-Based gate. The **random gate** outputs a confidence score drawn independently at each reasoning step. Specifically, at step i the gate produces $m_i \sim \text{Uniform}(0, 1)$.

The **probe-based gate** uses a learned linear transformation over a layer’s intermediate model representations to estimate whether further reasoning is necessary. Let $h_i^l \in \mathbb{R}^d$ denote the final token hidden state extracted from the model’s l^{th} layer at reasoning step i . The gate computes

$$m_i = \sigma(w^\top h_i^l + b), \quad (6)$$

where $w \in \mathbb{R}^d$, $b \in \mathbb{R}$ are learned parameters and $\sigma(\cdot)$ denotes the sigmoid function, ensuring $m_i \in (0, 1)$. The gate is trained to predict whether the predicted answer at step i matches the eventual final answer produced at step n . The supervision signal is derived post hoc from full reasoning traces and does not influence the generation process. As the gate is a simple linear transform over intermediate representations, it is computationally efficient and can be applied at inference time without modifying the underlying model architecture or generation procedure at negligible cost.

4.2 Experimental Setup

Reasoning trace collection. For each dataset D , we first collect full reasoning traces by running the model to completion. For each example, this yields a reasoning trace $R = (R_1, \dots, R_n)$. We then use forced answer completion to get an answer trajectory $T = (A_0, \dots, A_n)$, and the index of the final answer switch t^* as defined in Section 2.3.1. Collected traces are split at the example level into training, validation, and test sets. All probe parameters are learned using only the training split, with hyperparameters selected on the validation split. The test split is held out for final evaluation only.

Task-specific and generic probe-based gate.

We train probe-based gates under two settings. In the task-specific setting, we train a separate probe for each dataset using only that dataset’s training split, allowing the probe to specialize to dataset-specific reasoning dynamics. In addition, we train a generic probe on the union of training splits from all datasets. The motivation is to test whether there exists a shared, task-agnostic structure in the model’s intermediate representations that reliably signals when further reasoning is unnecessary. The generic probe is evaluated on each dataset’s test split without any dataset-specific adaptation. Comparing task-specific and generic probes, therefore, we examine whether early stopping can be governed by a universal signal, rather than task or dataset-specific heuristics. Further training details

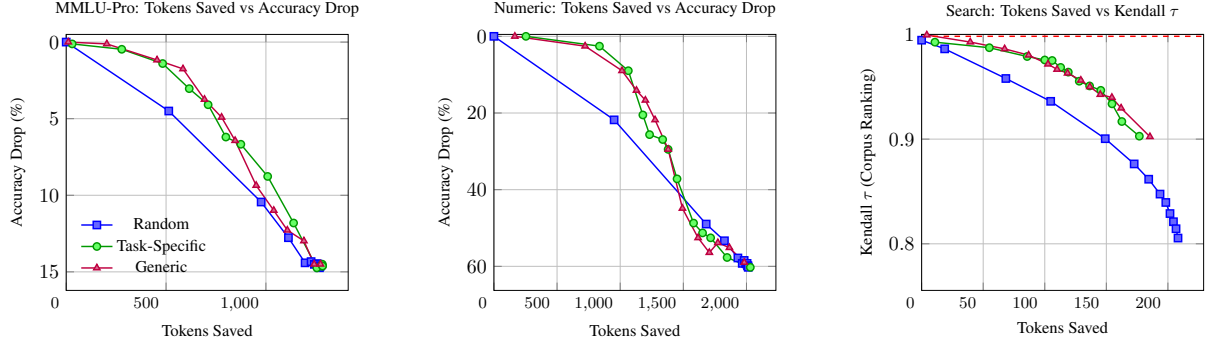


Figure 4: Early stopping performance for random, task-specific and generic probe-based gates on Qwen3-4B reasoning. We provide results on other models in Appendix G

are given in Appendix I.

4.3 Early Stopping Results

We plot the number of reasoning tokens saved and the resulting drop in final answer accuracy for MCQ and Numeric test sets in the left two of Figure 4, and the kendall-tau of the ranking of candidate documents using the intermediate query against the ranking using the final query for the Search test set in the right one. In tool-selection as the intermediate tool at different reasoning steps remains unchanged, early stopping induces no measurable degradation in task performance. For this reason, we do not report accuracy–token trade-off curves for tool-selection tasks, as the accuracy is effectively invariant to the stopping point under our evaluation protocol. We find that random, task-

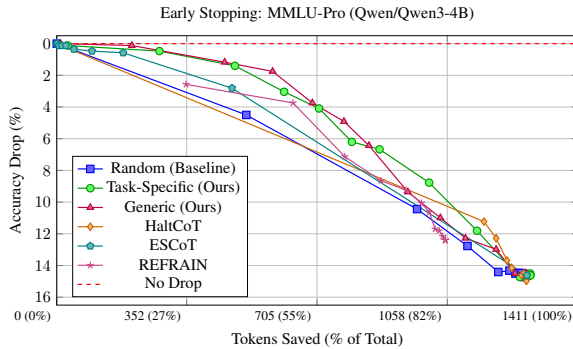


Figure 5: Comparison of our probe-based early stopping method with existing methods on the MMLU dataset (MCQ).

specific probe, and generic probe-based gates all **yield a smooth trade-off between token savings and final answer accuracy, with probe-based gates consistently outperforming random** gates across all tasks and models. We also compare our methods with existing methods such as Halt-

COT (Laaouach, 2025), ESCoT (Mao et al., 2026), and REFRAIN (Sun et al., 2025) in Figure 5. We find that **our probe-based gates achieve better token savings with less accuracy drop compared to existing methods especially in the $< 2\%$ accuracy drop regime**. Probe-based gates in both task-specific and generic settings can effectively **save 500 tokens with less than 2% accuracy drop** on Qwen3-4B. At higher savings of 1,000 tokens, generic probing on the numeric dataset still limits the accuracy drop to roughly 4%. Although numeric tasks are inherently more sensitive to truncated reasoning, probing continues to provide a markedly better trade-off, suggesting that answer determination often precedes the completion of full arithmetic derivations.

Generic probing preserves a kendall- τ of 0.97 while saving 100 tokens, and remains above **0.9 even when saving 200 tokens**. In contrast, random stopping drops below 0.9 at similar token savings. This demonstrates that probing-based early stopping preserves retrieval performance even when a large fraction of reasoning tokens is removed. These early-stopping results mirror our trajectory analysis. The post-decision tokens identified with our trajectory analysis can be removed at little cost. We observe no consistent trend with model size, with similar patterns across scales. Exhaustive results are reported in Appendix G.

Task-Specific vs Generic Probe-based gate

Across tasks, generic probe closely matches token–performance trade-off achieved by individually trained task-specific gates, with no consistent loss in token savings or final answer quality.

Despite being trained without any dataset-specific supervision, the generic probe gate yields early stopping frontiers that largely overlap with

those of the individual probes in Figure 4. This indicates that the intermediate representations used by the probe to decide when further reasoning is unnecessary capture signals that are largely shared across tasks. The comparable performance of the generic and individual gates therefore suggests the presence of a larger universal structure governing answer switches and early stopping, rather than task-specific stopping behavior.

We find that the accuracy under forced answer completion falls consistently between that of full reasoning and no reasoning, as shown in Figure 4. **This indicates that forced answer completion does not corrupt the model’s performance, but rather elicits the model’s decision early.**

5 Related Work

Faithfulness of Chain-of-Thought. Chain-of-thought (CoT) prompting improves performance on many reasoning tasks by encouraging models to generate intermediate steps before giving a final answer (Wei et al., 2022; Kojima et al., 2022). However, recent work questions whether these reasoning steps truly reflect how models reach their final answers. Prior studies show that models can produce correct answers even when their explanations are unnecessary, incorrect, misleading, or added after the fact (Wan et al., 2024; Wiegrefe and Pinter, 2019; Chiang and yi Lee, 2024; Paul et al., 2024; Arcuschin et al., 2025). Other work finds that reasoning traces can be edited, rewritten, or significantly shortened with little effect on accuracy, suggesting that not all generated reasoning is necessary for the final prediction (Zhou et al., 2023; Wang et al., 2022). Using reasoning step corruption, (Ye et al., 2026) show that after roughly 70-85% of the reasoning trace, model predictions become largely insensitive to subsequent reasoning steps, suggesting that much of the remaining trace functions as post hoc explanation rather than computation. Together, these results suggest that many reasoning tokens may not directly influence the model’s decision.

Efficient Reasoning. As reasoning-enabled models become more widely deployed, the computational cost of long reasoning traces has emerged as a practical concern. Verbose reasoning increases inference latency, memory usage, and energy consumption, and can complicate deployment in tool-augmented or agentic systems (Yao et al., 2022; Karpas et al., 2022). Several recent works have

therefore explored methods for reducing reasoning cost while preserving performance. These include approaches that encourage models to produce shorter or more selective chains of thought (Zhou et al., 2023; Paul et al., 2024). Other work targets the cost of self-consistency (Wang et al., 2022), which boosts accuracy by majority-voting over multiple sampled traces but multiplies inference cost accordingly. Proposed remedies include halting low-quality generations with a reward model (Sun et al., 2024) or splitting the sample budget into sequential windows (Li et al., 2024). There is work that also optimizes the model’s performance at various different reasoning truncation levels (Qi et al., 2025). Related work has also explored efficiency-oriented early stopping strategies that aim to balance reasoning depth against computational cost (Laaouach, 2025; Sun et al., 2025; Mao et al., 2026; Kang et al., 2025). However, these approaches typically rely on heuristic stopping criteria that often fail to generalise beyond the task settings in which they were designed. In particular, many are tailored to constrained evaluation formats such as multiple-choice reasoning, while others struggle to reliably determine whether the model’s intermediate reasoning or predicted answer has genuinely stabilised. There exists work that also tries to predict the success of a chain-of-thought as it is being generated, and halts generation if the chain is predicted to be unsuccessful (Afzal et al., 2025). However, this would require restarting the reasoning process from scratch.

Answer Dynamics. A smaller line of work has investigated intermediate predictions during generation, often in the context of uncertainty estimation (Mei et al., 2025; Yoon et al., 2025; Zhang and Zhang, 2025; Mao and Venkat, 2026), or self-consistency (Wang et al., 2022; Liang et al., 2024). These methods implicitly acknowledge that a model’s belief about the correct answer may evolve, but they do not directly characterize the temporal structure of answer changes within a single reasoning trace. Another work tries to decode answers from the model’s residual stream before the Chain of Thought occurs (Cox et al., 2026). Our work builds on these insights by explicitly tracing model decisions at intermediate reasoning steps.

Unlike prior work, which treats reasoning as a static object and asks whether an explanation is faithful, we measure *when* the decision forms, using forced answer completion to build per-step

answer trajectories. And rather than relying on surface-level heuristics, we learn the stopping signal using the model’s hidden states, studied across multiple task formats rather than one.

6 Conclusion

We used forced answer completion to track when a reasoning model commits to its answer. On Qwen3-4B, the predicted answer changes in only 32% of queries, and when it does the switch usually happens early. Otherwise it matches the no-reasoning baseline from the start. Across datasets, roughly half of all reasoning tokens, an average of 760, are produced after the final switch without altering the predicted answer. This gap between when the answer settles and when generation stops is what early stopping exploits. A linear probe over intermediate hidden states saves about 500 tokens per query on Qwen3-4B at under a 2% accuracy drop, and a single generic probe trained on all four tasks performs on par with the per-task probes. While the linear probe is unlikely to generalize completely out-of-distribution, it achieves strong in-distribution fit, absorbing the stopping criteria of several distinct reasoning regimes within a single set of parameters.

7 Limitations

Despite its promising results, our study is limited to the models, and tasks evaluated in this work. We do not study substantially larger models or much different task scenarios, where answer dynamics may differ. In addition, our experiments focus on language based reasoning tasks, and we leave extensions to broader multimodal models for future work.

8 Ethical Considerations

This work studies inference-time efficiency for reasoning models and does not involve human subjects, private data, or the release of new models. All experiments use publicly available models and datasets, employed in line with their intended research use. Our early-stopping methods reduce token usage, inference cost, and the associated energy and carbon footprint of deploying reasoning models, which we view as a positive outcome. At the same time, halting generation once an answer appears stable trades a small amount of accuracy for efficiency, and in high-stakes domains such as medical or legal reasoning this trade-off should be applied with care, since truncated reasoning could

omit verification steps even when the surfaced answer is unchanged. Our analysis also concerns only the answer a model surfaces, not its internal reasoning faithfulness, so our findings should not be read as claims about whether a model’s stated rationale reflects its underlying computation. All the models used in this work are publicly available on Huggingface and free for research. The datasets are licensed under Creative Commons by Non-Commercial 4.0 (CC by NC 4.0). No anticipated risks are associated with using the data from these provided links.

References

- Anum Afzal, Florian Matthes, Gal Chechik, and Yftah Ziser. 2025. [Knowing before saying: Llm representations encode information about chain-of-thought success before completion](#). *Preprint*, arXiv:2505.24362.
- Iván Arcuschin, Jett Janiak, Robert Krzyzanowski, Senthoooran Rajamanoharan, Neel Nanda, and Arthur Conmy. 2025. [Chain-of-thought reasoning in the wild is not always faithful](#). *Preprint*, arXiv:2503.08679.
- Mislav Balunovic, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. 2026. Matharena: Evaluating llms on uncontaminated math competitions. *Advances in Neural Information Processing Systems*, 38.
- Center for AI Safety, Scale AI, and HLE Contributors Consortium. 2026. [A benchmark of expert-level academic questions to assess AI capabilities](#). *Nature*, 649:1139–1146.
- Cheng-Han Chiang and Hung yi Lee. 2024. [Over-reasoning and redundant calculation of large language models](#). *Preprint*, arXiv:2401.11467.
- Kyle Cox, Darius Kianersi, and Adrià Garriga-Alonso. 2026. [Decoding answers before chain-of-thought: Evidence from pre-cot probes and activation steering](#). *Preprint*, arXiv:2603.01437.
- Liwei Kang, Yue Deng, Yao Xiao, Zhanfeng Mo, Wee Sun Lee, and Lidong Bing. 2025. [First try matters: Revisiting the role of reflection in reasoning models](#). *Preprint*, arXiv:2510.08308.
- Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, and 1 others. 2022. Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv preprint arXiv:2205.00445*.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in*

- neural information processing systems*, 35:22199–22213.
- Yassir Laouach. 2025. [HALT-cot: Model-agnostic early stopping for chain-of-thought reasoning via answer entropy](#). In *4th Muslims in ML Workshop co-located with ICML 2025*.
- Andrew Lampinen, Ishita Dasgupta, Stephanie Chan, Kory Mathewson, Mh Tessler, Antonia Creswell, James McClelland, Jane Wang, and Felix Hill. 2022. Can language models learn from explanations in context? In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 537–563.
- Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, and 1 others. 2023. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*.
- Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857.
- Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024. [Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning](#). *Preprint*, arXiv:2401.10480.
- Xun Liang, Shichao Song, Zifan Zheng, Hanyu Wang, Qingchen Yu, Xunkai Li, Rong-Hua Li, Yi Wang, Zhonghao Wang, Feiyu Xiong, and 1 others. 2024. Internal consistency and self-feedback in large language models: A survey. *arXiv preprint arXiv:2407.14507*.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*.
- Minjia Mao, Bowen Yin, Yu Zhu, and Xiao Fang. 2026. [Early stopping chain-of-thoughts in large language models](#).
- Zhenjiang Mao and Anirudh Venkat. 2026. Recurrent confidence chain: Temporal-aware uncertainty quantification in large language models. *arXiv preprint arXiv:2601.13368*.
- Zhiting Mei, Christina Zhang, Tenny Yin, Justin Lillard, Ola Shorinwa, and Anirudha Majumdar. 2025. Reasoning about uncertainty: Do reasoning models know when they don’t know? *arXiv preprint arXiv:2506.18183*.
- Joao Monteiro, Pierre-Andre Noel, Etienne Marcotte, Sai Rajeswar, Valentina Zantedeschi, David Vazquez, Nicolas Chapados, Christopher Pal, and Perouz Taslakian. 2024. [Replika: A question-answering dataset for benchmarking llms on unseen reference content](#). *Preprint*, arXiv:2406.11811.
- OpenAI. 2025. [gpt-oss-120b & gpt-oss-20b model card](#). *Preprint*, arXiv:2508.10925.
- Debjit Paul, Robert West, Antoine Bosselut, and Boi Faltings. 2024. [Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning](#). *Preprint*, arXiv:2402.13950.
- Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki Van Stein, and Thomas Bäck. 2025. Multi-step reasoning with large language models, a survey. *ACM Computing Surveys*, 58(6):1–35.
- Penghui Qi, Zichen Liu, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. [Optimizing any-time reasoning via budget relative policy optimization](#). *Preprint*, arXiv:2505.13438.
- QuotientAI. 2025. [Mcp tool call evaluation test dataset](#).
- David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. [Gpqa: A graduate-level google-proof q&a benchmark](#). *Preprint*, arXiv:2311.12022.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. 2024. [Fast best-of-n decoding via speculative rejection](#). *Preprint*, arXiv:2410.20290.
- Renliang Sun, Wei Cheng, Dawei Li, Haifeng Chen, and Wei Wang. 2025. [Stop when enough: Adaptive early-stopping for chain-of-thought reasoning](#). *Preprint*, arXiv:2510.10103.
- Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, and 304 others. 2026. [Gemma 4 technical report](#). *Preprint*, arXiv:2607.02770.
- Alexander Wan, Eric Wallace, and Dan Klein. 2024. What evidence do language models find convincing? *arXiv preprint arXiv:2402.11782*.

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

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhuranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. 2024. [Mmlu-pro: A more robust and challenging multi-task language understanding benchmark](#). *Preprint*, arXiv:2406.01574.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Sarah Wiegrefe and Yuval Pinter. 2019. Attention is not not explanation. *arXiv preprint arXiv:1908.04626*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Donald Ye, Max Loffgren, Om Kotadia, and Linus Wong. 2026. [Mechanistic evidence for faithfulness decay in chain-of-thought reasoning](#). *Preprint*, arXiv:2602.11201.

Dongkeun Yoon, Seungone Kim, Sohee Yang, Sunkyoung Kim, Soyeon Kim, Yongil Kim, Eunbi Choi, Yireun Kim, and Minjoon Seo. 2025. Reasoning models better express their confidence. *arXiv preprint arXiv:2505.14489*.

Boxuan Zhang and Ruqi Zhang. 2025. Cot-ug: Improving response-wise uncertainty quantification in llms with chain-of-thought. *arXiv preprint arXiv:2502.17214*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. [Least-to-most prompting enables complex reasoning in large language models](#). In *The Eleventh International Conference on Learning Representations*.

A Datasets

We use the sample sizes described in Table 2 to conduct our analyses and early stopping experiments for all the datasets. We shuffle and randomly sample from the dataset to obtain samples.

Dataset	S
MCQ	1000
Numeric-answer	500
Search-query	1000
Tool-selection	1000

Table 2: Sample Sizes of reasoning traces

B Answer Prefix

We choose an answer prefix P that elicits a probability distribution over the task answer space from the model’s vocabulary-level output distribution. The choice of P depends on the task format. Below, we describe the prefixes used for each class of task considered in this work.

B.1 MCQ Answer Prefix

For multiple-choice questions (MCQs), we instruct the model to respond in a fixed JSON format: `{"answer": "A"}`, where A can be replaced by any valid answer label. During forced decoding, we set the answer prefix to $P = \{"answer": "$, which induces a distribution over the possible MCQ answer labels at the next token position. We intervene at each reasoning step and record the unnormalized logits corresponding to the candidate answer labels.

Component	Value
Expected output format	<code>{"answer": "A"}</code>
Answer prefix P	<code>{"answer": "</code>
Predicted token	A, B, C or D

Table 3: Answer prefix used for MCQ tasks.

B.2 Numeric Answer Prefix

For tasks whose answers are numeric, we instruct the model to output its answer inside `\boxed{}` tags, e.g., `\boxed{42}`. During forced decoding, we set $P = \boxed{$ to elicit a distribution over possible numeric answers. We restrict our analysis to questions whose correct answer is represented by a single token in the model vocabulary.

Component	Value
Expected output format	<code>\boxed{42}</code>
Answer prefix P	<code>\boxed{</code>
Predicted token	Single-token numeric answer

Table 4: Answer prefix used for numeric-answer tasks.

B.3 Tool Selection Prefix

For user requests where the model must select exactly one tool from a provided list, we track the evolution of the selected tool across reasoning steps. To simplify the setup, we consider tools without arguments. We use the model’s native tool-calling format and choose a prefix such that the next generated token corresponds to the tool name.

For models using Hermes-style function calling (e.g., Qwen3), the format is:

Prefix Format
<code>C<think>R₁R₂. . . R_k</think></code> <code><tool_call>{"name": "</code>

Table 5: Answer prefix used for tool selection tasks. The highlighted token corresponds to the predicted tool name.

B.4 Tool Argument Prefix

For tasks where the model is forced to call a specific tool that takes a single argument (e.g., a web search tool), we track the evolution of the generated argument across reasoning steps. Generation proceeds using the same sampling settings and is terminated at the closing tool call tag (`</tool_call>`).

The prefix used to elicit the argument token is shown below.

Prefix Format
<code>C<think>R₁R₂. . . R_k</think></code> <code><tool_call>{"name": "web_search",</code> <code>"arguments": {"query": "</code>

Table 6: Answer prefix used for tool argument generation. The highlighted token corresponds to the generated argument.

C Search-Query Implementation Details

In this section we describe the implementation details of the search-query task, including how we extract the generated query text during forced answer completion and how we compare it to the query

produced under full generation using the answer label mapping rules. When forced answer completion causes the model to emit a search query, we extract the query text and compare it to the query produced under full generation using the answer label mapping rules. We do not execute the search; instead, we compare the generated queries directly and use query similarity as a proxy for retrieval equivalence.

During forced answer completion, we allow the model to fully generate the search query using the same sampling settings as used for the reasoning trace. To compare two queries obtained at reasoning steps i and j , denoted A_i and A_j , we embed each query using a fixed query encoder to obtain vector representations $\mathbf{a}_i$ and $\mathbf{a}_j$. We consider the two queries to be equivalent if their cosine similarity exceeds a fixed threshold.

$$A_i = A_j \iff \frac{\mathbf{a}_i^\top \mathbf{a}_j}{\|\mathbf{a}_i\| \|\mathbf{a}_j\|} \geq \gamma, \quad (7)$$

where $\gamma = 0.9$ in all experiments.

D Answer Trajectory Denoising

A high frequency of TAFs indicates that $A_i \neq A_{i+\ell}$ for a small number of further reasoning steps ℓ , despite $A_i = A_{i+\ell+1}$. That is, the model briefly deviates from an answer and then quickly returns to its previous value. Such short-lived reversals are unlikely to correspond to changes in the model’s final decision, since they do not persist long enough to affect the eventual output. Instead, they typically reflect local instability in the intermediate reasoning trace, arising from transient token-level fluctuations or sensitivity to nearby context, rather than deliberate revisions of the underlying conclusion.

Motivated by this observation, we aim to filter out these short-lived deviations in the answer trajectory and retain only answer changes that are sustained over multiple consecutive reasoning steps. To this end, we apply a simple causal smoothing procedure that suppresses brief oscillations while preserving persistent answer switches. This denoising removes excess answer switches that do not meaningfully correspond to decision changes, but instead arise from momentary answer flips. As a result, the estimated final answer switch becomes less sensitive to noisy local fluctuations, revealing that even more post-decision reasoning tokens can be saved once such spurious flips are discounted.

Hold-for- k Smoothing Given an answer trajectory $T = (A_0, A_1, A_2, \dots, A_n)$, we define a causal

smoothing operator, referred to as *hold-for- k* . The operator enforces persistence by requiring an answer to remain unchanged for k consecutive reasoning steps before it is accepted as a genuine answer switch. Intuitively, this prevents brief, noisy answer flips from being treated as meaningful changes.

Formally, the smoothed answer trajectory $\tilde{T}^{(k)} = (\tilde{A}_1^{(k)}, \dots, \tilde{A}_n^{(k)})$ is defined recursively as:

$$\tilde{A}_i^{(k)} = \begin{cases} A_i, & \text{if } A_{i-j} = A_i \quad \forall j \in \{0, \dots, k-1\}, \\ \tilde{A}_{i-1}^{(k)}, & \text{otherwise.} \end{cases} \quad (8)$$

This construction is causal, as the smoothed answer at step i depends only on the current and previous reasoning steps and does not rely on future information. This property is essential for online or streaming settings, where answer trajectories are observed incrementally as the model generates its reasoning. In such settings, non-causal smoothing would require access to future steps and therefore cannot be applied in real time. By enforcing causality, hold-for- k smoothing enables on-the-fly monitoring of answer trajectories, allowing sustained answer switches to be detected as they emerge during generation rather than retrospectively.

We sweep values of k and measure the percentage of answer switches in Qwen3-4B MMLU-pro reasoning traces that are classified as transient answer flips in Figure 6. We find that values of $k \geq 3$ give diminishing amounts of switches classified to be transient. We hence choose $k = 3$ for the rest of our experiments.

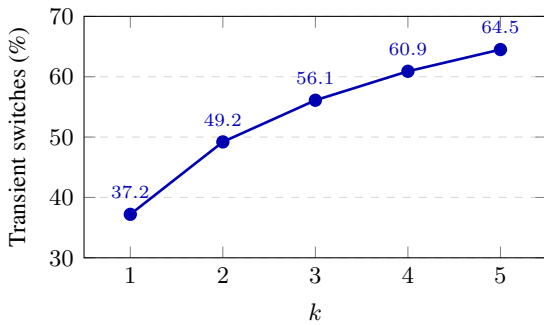


Figure 6: Percentage of Answer Switches that are transient and the values of k in our denoising method for Qwen3-4B MMLU-pro reasoning traces.

E Effect of Denoising

We evaluate the effect of answer trajectory denoising on the structure of predicted answer sequences. The denoising procedure is applied to suppress transient answer flips while preserving sustained an-

swer changes. Figure 3 compares the original and denoised answer trajectory of an example: Short-lived answer changes present in the raw trajectory are removed after denoising, while longer-lasting answer changes are retained.

We also show the effect of denoising at the dataset level in the distribution plots in Appendix H. Compared to the original distribution, the denoised distribution exhibits fewer examples with a large number of answer switches, and the mass is concentrated on smaller switch counts. For example, as seen in Table 1, the average answer switches decrease when the trajectory is denoised, $\approx 8 \rightarrow \approx 2$ **for the Numeric-answer dataset across all models**.

These results show that denoising removes transient answer flips and yields answer switch distributions that are less spread out, while preserving the overall structure of sustained answer changes.

F Experiments on Frontier Reasoning Datasets

We extend our analysis to substantially harder reasoning benchmarks to evaluate whether the observed answer trajectory dynamics persist in more challenging settings. Specifically, we consider Humanity’s Last Exam (Center for AI Safety et al., 2026), GPQA-Diamond (Rein et al., 2023), and AIME 2026 problems (Balunovic et al., 2026), which are designed to require deeper multi-step reasoning and exhibit lower overall accuracy compared to the datasets considered earlier and evaluate them on Qwen3-4B.

From Table 7, across all three benchmarks, we observe qualitatively consistent behavior with our main results. In particular, the predicted answer often stabilizes before the completion of the full reasoning trace, even in these more demanding settings. Although these datasets exhibit higher answer-switch counts and require longer reasoning chains, the final answer switch typically occurs well before the end of generation.

We measure the number of reasoning tokens generated after the final answer switch as defined in Section 3.5. Despite the increased task difficulty, a substantial fraction of reasoning tokens are generated after the predicted answer has already stabilized. On average, we find that approximately 32%–67% of the total reasoning tokens are produced after the final answer switch across these datasets.

These results indicate that the temporal gap between decision formation and subsequent reasoning is not limited to moderate-difficulty or proxy tasks, but persists even in substantially harder reasoning regimes. This further supports the generality of the early decision-making phenomenon observed in our study.

G Full Early Stopping Results

We evaluate early stopping methods on Qwen/Qwen3-4B, Qwen/Qwen3-8B, Qwen/Qwen3-14B, and Qwen/Qwen3-30B-A3B.

G.1 Qwen/Qwen3-4B

Tables 8-16 show the results of early stopping experiments for Qwen/Qwen3-4B

G.1.1 Random Gate

Table 8: MCQ

Accuracy drop	Tokens saved	Token %
14.620	1273	99.2
14.596	1271	99.0
14.725	1268	98.8
14.678	1264	98.5
14.480	1259	98.1
14.468	1253	97.6
14.515	1243	96.8
14.327	1226	95.5
14.409	1196	93.2
12.772	1113	86.7
10.444	976	76.1
4.503	513	40.0
0.000	0	0.0

Table 9: Numeric-answer

Accuracy drop	Tokens saved	Token %
60.256	2014	99.2
59.872	2012	99.1
59.872	2008	98.9
59.231	2005	98.7
59.359	2000	98.5
59.231	1993	98.2
58.462	1981	97.6
59.231	1966	96.8
57.821	1932	95.2
53.333	1825	89.9
48.974	1681	82.8
21.795	952	46.9
0.000	0	0.0

Table 10: Search-query

Kendall τ	Tokens saved	Token %
0.805	208	91.0
0.814	207	90.3
0.821	205	89.4
0.828	202	88.2
0.839	198	86.7
0.847	194	84.7
0.861	184	80.6
0.876	173	75.5
0.900	149	65.2
0.936	105	45.8
0.957	68	29.9
0.986	19	8.2
0.994	0	0.0

G.1.2 Task-specific probing

Table 11: MCQ

Accuracy drop	Tokens saved	Token %
14.620	1283	100.0
14.620	1283	100.0
14.503	1283	99.9
14.737	1255	97.8
11.813	1139	88.7
8.772	1009	78.6
6.667	874	68.1
6.199	800	62.3
4.094	711	55.4
3.041	616	48.0
1.404	483	37.6
0.468	278	21.7
0.117	30	2.3

Table 7: Summary of answer trajectory metrics on additional datasets.

Dataset	Accuracy (%)	$\Pr(t^* = -1)$	AnswerSwitches Raw / Denoised	TAFs(T)	T_{after}
		Raw% / Denoised%			Raw / Denoised
Humanity’s Last Exam	5.2	39.2 / 48.9	10.8 / 1.3	6.5	1186 (49.9%) / 1765 (67.1%)
GPQA Diamond	56.1	35.5 / 43.8	10.9 / 1.5	6.6	1608 (45.7%) / 2259 (59.8%)
AIME26	72.7	27.2 / 27.2	14.3 / 2.1	7.1	660 (16.4%) / 1654 (32.9%)

Table 12: Numeric-answer

Accuracy drop	Tokens saved	Token %
60.256	2030	100.0
57.692	1846	90.9
52.564	1716	84.5
51.282	1652	81.4
48.718	1581	77.9
37.179	1450	71.4
29.487	1380	68.0
26.923	1336	65.8
25.641	1233	60.7
20.513	1179	58.1
8.974	1064	52.4
2.564	836	41.2
0.000	254	12.5

G.1.3 General probing

Table 14: MCQ

Accuracy drop	Tokens saved	Token %
14.503	1273	99.2
15.088	1238	96.4
14.620	1171	91.3
13.918	1096	85.4
10.643	1025	79.8
9.006	935	72.9
6.901	818	63.8
5.848	736	57.3
2.924	643	50.1
1.988	533	41.5
0.585	353	27.5
0.468	142	11.1
0.117	8	0.6

Table 13: Search-query

Kendall τ	Tokens saved	Token %
0.902	176	77.3
0.916	162	71.1
0.933	154	67.6
0.946	145	63.6
0.950	136	59.6
0.955	128	55.9
0.964	119	52.0
0.968	112	49.3
0.975	106	46.3
0.975	99	43.7
0.975	85	37.4
0.987	54	24.0
0.992	10	4.7

Table 15: Numeric-answer

Accuracy drop	Tokens saved	Token %
56.410	1940	95.6
58.974	1838	90.6
58.974	1746	86.0
55.128	1658	81.7
48.718	1572	77.4
42.308	1484	73.1
30.769	1298	63.9
28.205	1249	61.5
25.641	1146	56.4
19.231	1048	51.6
7.692	861	42.4
1.282	502	24.7
0.000	25	1.2

Table 16: Search-query

Kendall τ	Tokens saved	Token %
0.902	185	81.0
0.929	162	70.8
0.939	154	67.5
0.942	145	63.4
0.949	136	59.7
0.956	129	56.5
0.963	118	51.8
0.966	109	48.0
0.971	102	44.8
0.980	86	38.0
0.986	67	29.4
0.992	39	17.2
0.999	4	0.6

Table 18: Numeric-answer

Accuracy drop	Tokens saved	Token %
59.722	2920	99.3
59.306	2918	99.3
59.028	2914	99.1
59.306	2910	99.0
59.167	2905	98.8
58.611	2898	98.6
58.333	2886	98.2
59.444	2872	97.7
58.750	2839	96.6
55.000	2734	93.0
50.556	2559	87.1
24.792	1588	54.0
0.000	0	0.0

Table 19: Search-query

Kendall τ	Tokens saved	Token %
0.840	257	92.5
0.845	256	92.0
0.849	254	91.2
0.855	251	90.2
0.861	247	89.0
0.867	242	87.2
0.873	233	83.7
0.881	220	79.2
0.901	196	70.4
0.929	139	50.0
0.950	94	33.9
0.981	25	8.9
0.990	0	0.0

G.2 Qwen/Qwen3-8B

Tables 17-25 show the results of early stopping experiments for Qwen/Qwen3-8B. We plot these results in Figure 7.

G.2.1 Random Gate

Table 17: MCQ

Accuracy drop	Tokens saved	Token %
24.390	1764	99.5
24.429	1762	99.4
24.527	1759	99.2
24.527	1755	99.0
24.473	1751	98.8
24.571	1745	98.4
24.732	1735	97.9
24.707	1721	97.0
24.141	1690	95.3
22.459	1608	90.7
19.107	1468	82.8
9.112	854	48.2
0.488	0	0.0

Table 20: MCQ

Accuracy drop	Tokens saved	Token %
23.415	1763	99.4
25.366	1673	94.4
24.878	1617	91.2
25.854	1552	87.5
20.976	1457	82.2
14.634	1350	76.1
10.244	1211	68.3
6.829	1100	62.1
3.902	993	56.0
0.976	843	47.5
1.951	545	30.7
0.488	279	15.7
0.000	34	1.9

G.2.2 Task-specific probing

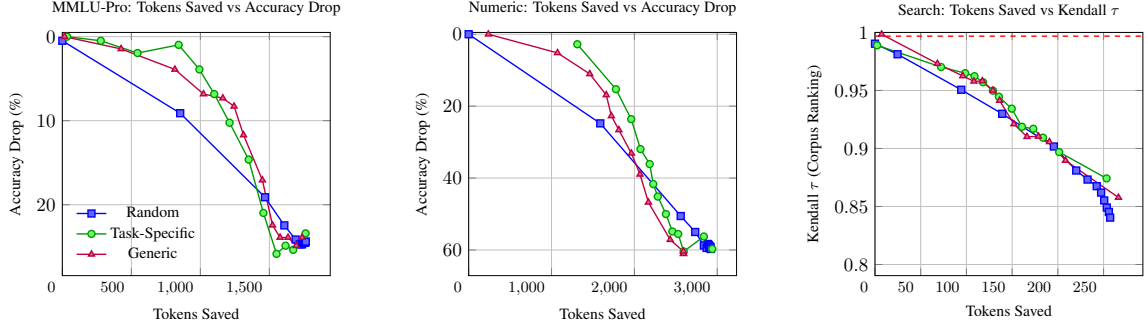


Figure 7: Early stopping performance for random, task-specific and generic probe-based gates on Qwen3-8B reasoning.

Table 21: Numeric-answer

Accuracy drop	Tokens saved	Token %
59.722	2939	100.0
56.250	2839	96.6
60.417	2595	88.3
55.556	2528	86.0
54.861	2458	83.6
50.000	2380	81.0
45.139	2282	77.6
41.667	2228	75.8
36.111	2185	74.3
31.944	2073	70.5
23.611	1962	66.7
15.278	1777	60.5
2.778	1311	44.6

G.2.3 General probing

Table 23: MCQ

Accuracy drop	Tokens saved	Token %
23.902	1737	98.0
24.878	1695	95.6
23.902	1635	92.2
23.902	1578	89.0
22.439	1523	85.9
17.073	1449	81.7
11.707	1313	74.1
8.293	1244	70.2
7.317	1163	65.6
6.829	1023	57.7
3.902	815	46.0
1.463	425	24.0
0.000	16	0.9

Table 22: Search-query

Kendall τ	Tokens saved	Token %
0.874	253	91.1
0.896	201	72.4
0.909	183	66.1
0.917	173	62.2
0.918	160	57.8
0.934	149	53.8
0.944	135	48.7
0.949	129	46.4
0.957	118	42.5
0.962	108	39.1
0.964	98	35.4
0.970	72	26.0
0.988	2	0.8

Table 24: Numeric-answer

Accuracy drop	Tokens saved	Token %
61.039	2593	100.0
61.039	2593	100.0
60.390	2591	99.9
57.143	2432	93.8
46.753	2169	83.7
38.961	2068	79.8
33.117	1964	75.7
26.623	1813	69.9
22.727	1722	66.4
16.883	1660	64.0
11.039	1462	56.4
5.195	1072	41.4
0.000	238	9.2

Table 25: Search-query

Kendall τ	Tokens saved	Token %
0.857	266	95.7
0.889	208	74.9
0.905	2190	68.6
0.910	178	64.2
0.910	166	59.7
0.921	151	54.6
0.941	136	49.0
0.949	129	46.4
0.958	117	42.2
0.957	108	38.9
0.962	96	34.5
0.973	68	24.5
0.998	7	2.6

Table 27: Numeric-answer

Accuracy drop	Tokens saved	Token %
54.135	2872	99.3
54.248	2869	99.2
54.060	2866	99.0
53.647	2861	98.9
54.023	2856	98.7
54.361	2850	98.5
53.947	2838	98.1
54.323	2820	97.5
53.195	2782	96.2
50.827	2675	92.5
44.887	2487	86.0
19.586	1462	50.5
0.000	0	0.0

Table 28: Search-query

Kendall τ	Tokens saved	Token %
0.836	201	90.4
0.840	200	89.7
0.845	198	88.8
0.850	195	87.5
0.856	191	85.8
0.863	186	83.5
0.873	178	79.8
0.886	165	73.9
0.906	142	63.6
0.935	100	44.8
0.957	61	27.2
0.983	14	6.4
0.990	0	0.0

G.3 Qwen/Qwen3-14B

Tables 26-34 show the results of early stopping experiments for Qwen/Qwen3-14B. We also plot the results in Figure 8.

G.3.1 Random Gate

Table 26: MCQ

Accuracy drop	Tokens saved	Token %
17.431	1395	99.4
17.294	1393	99.2
17.431	1390	99.0
17.798	1386	98.7
17.752	1382	98.4
17.844	1376	98.0
17.569	1365	97.2
17.294	1351	96.3
17.477	1320	94.1
16.193	1235	88.0
13.991	1106	78.8
5.780	605	43.1
0.459	0	0.0

Table 29: MCQ

Accuracy drop	Tokens saved	Token %
16.972	1397	99.6
17.890	1364	97.2
17.890	1319	94.0
17.431	1265	90.1
14.679	1193	85.0
13.303	1118	79.6
11.009	1040	74.1
8.716	951	67.8
8.257	829	59.1
4.587	698	49.7
4.587	540	38.5
3.670	335	23.9
0.459	68	4.9

G.3.2 Task-specific probing

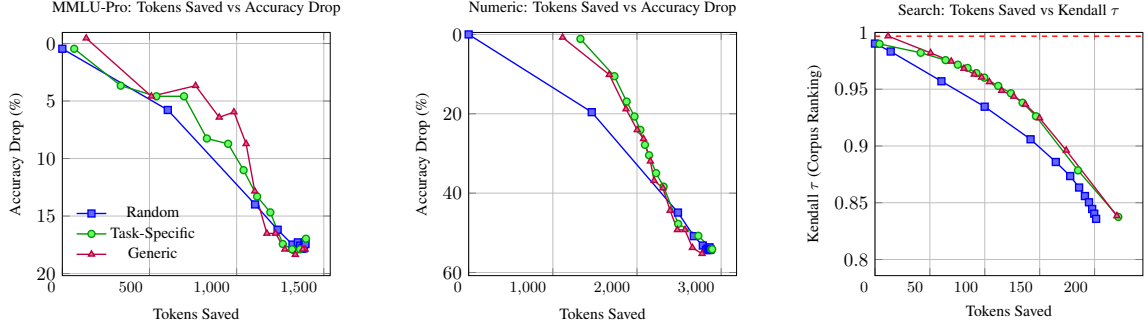


Figure 8: Early stopping performance for random, task-specific and generic probe-based gates on Qwen3-14B reasoning.

Table 30: Numeric-answer

Accuracy drop	Tokens saved	Token %
54.135	2893	100.0
54.135	2890	99.9
50.752	2729	94.3
47.744	2489	86.0
38.346	2316	80.0
34.962	2227	77.0
30.451	2143	74.1
27.820	2094	72.4
24.060	2040	70.5
20.677	1970	68.1
16.917	1877	64.9
10.526	1732	59.9
1.128	1328	45.9

G.3.3 General probing

Table 32: MCQ

Accuracy drop	Tokens saved	Token %
17.890	1392	99.2
17.890	1379	98.3
18.349	1337	95.2
17.890	1277	91.0
16.514	1223	87.2
16.514	1174	83.6
12.844	1104	78.7
8.716	1054	75.1
5.963	984	70.1
6.422	900	64.1
3.670	764	54.5
4.587	511	36.4
-0.459	137	9.8

Table 31: Search-query

Kendall τ	Tokens saved	Token %
0.837	222	99.5
0.878	185	83.0
0.926	147	65.8
0.938	134	60.3
0.946	124	55.5
0.953	112	50.3
0.960	100	44.7
0.964	92	41.5
0.969	84	37.8
0.972	75	33.9
0.976	64	28.8
0.982	42	18.7
0.990	4	1.8

Table 33: Numeric-answer

Accuracy drop	Tokens saved	Token %
55.263	2773	95.8
53.759	2659	91.9
49.248	2570	88.8
49.248	2480	85.7
44.361	2393	82.7
38.722	2308	79.8
36.842	2203	76.2
31.955	2159	74.6
26.316	2078	71.8
24.060	2002	69.2
18.797	1869	64.6
10.150	1671	57.8
0.752	1116	38.6

Table 34: Search-query

Kendall τ	Tokens saved	Token %
0.838	220	98.9
0.896	174	78.2
0.925	150	67.3
0.937	137	61.5
0.944	126	56.8
0.949	115	51.8
0.956	104	46.8
0.960	97	43.6
0.963	90	40.6
0.968	81	36.4
0.975	70	31.2
0.982	51	22.7
0.997	12	5.3

Table 36: Numeric-answer

Accuracy drop	Tokens saved	Token %
50.980	2975	99.2
50.588	2972	99.2
51.307	2969	99.0
50.719	2965	98.9
51.438	2960	98.7
51.111	2953	98.5
50.523	2940	98.1
50.654	2925	97.6
50.000	2889	96.4
46.471	2770	92.4
40.588	2599	86.7
18.627	1597	53.3
0.000	0	0.0

Table 37: Search-query

Kendall τ	Tokens saved	Token %
0.824	200	89.9
0.829	199	89.2
0.832	197	88.4
0.837	194	87.2
0.840	191	85.7
0.848	186	83.6
0.857	177	79.5
0.867	166	74.3
0.886	144	64.5
0.918	101	45.4
0.945	63	28.5
0.977	16	7.3
0.986	0	0.0

G.4 Qwen/Qwen3-30B-A3B

Tables 35-43 show the results of early stopping experiments for Qwen/Qwen3-30B-A3B. We also plot the results in Figure 9.

G.4.1 Random Gate

Table 35: MCQ

Accuracy drop	Tokens saved	Token %
17.021	1043	99.1
17.021	1041	98.8
17.074	1038	98.5
16.915	1034	98.2
16.809	1030	97.8
17.181	1024	97.2
16.968	1012	96.1
16.596	998	94.8
15.745	968	92.0
14.202	884	83.9
12.021	765	72.6
4.734	360	34.2
0.000	0	0.0

Table 38: MCQ

Accuracy drop	Tokens saved	Token %
10.106	815	77.3
5.319	658	62.5
2.128	491	46.6
1.064	414	39.3
0.532	342	32.5
0.532	247	23.5
1.064	162	15.4
0.532	124	11.8
0.000	103	9.7
0.000	53	5.1
0.000	34	3.3
0.000	17	1.7
0.000	7	0.6

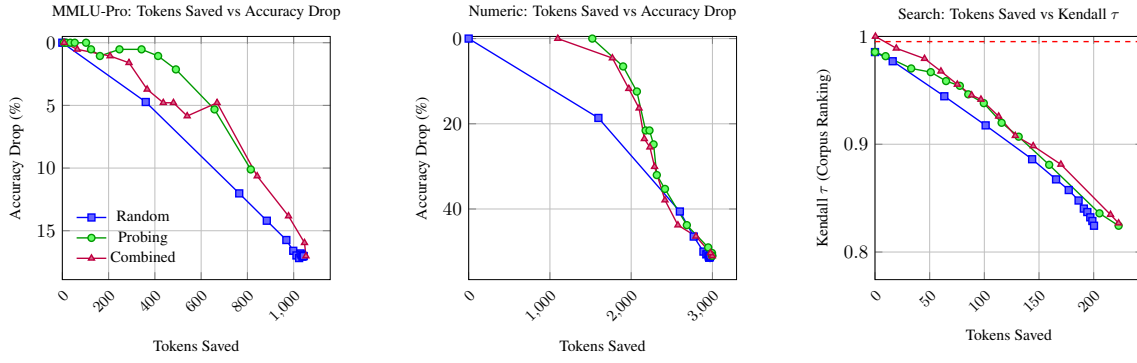


Figure 9: Early stopping performance for random, task-specific and generic probe-based gates on Qwen3-30B-A3B reasoning.

Table 39: Numeric-answer

Accuracy drop	Tokens saved	Token %
50.980	2997	100.0
50.980	2997	100.0
50.327	2995	99.9
49.020	2948	98.3
43.791	2685	89.6
35.294	2416	80.6
32.026	2315	77.2
24.837	2276	75.9
21.569	2225	74.2
21.569	2178	72.7
12.418	2070	69.1
6.536	1901	63.4
0.000	1521	50.8

G.4.3 General probing

Table 41: MCQ

Accuracy drop	Tokens saved	Token %
17.021	1053	100.0
15.957	1047	99.4
13.830	978	92.8
10.638	842	80.0
4.787	669	63.5
5.851	540	51.2
4.787	480	45.6
4.787	436	41.4
3.723	367	34.8
1.596	288	27.4
1.064	206	19.6
0.532	64	6.1
0.000	8	0.7

Table 40: Search-query

Kendall τ	Tokens saved	Token %
0.824	223	100.0
0.836	205	92.2
0.881	159	71.5
0.907	131	59.0
0.920	116	52.0
0.938	99	44.6
0.946	85	38.3
0.954	78	34.8
0.959	65	29.2
0.967	51	22.9
0.970	33	14.9
0.982	10	4.4
0.986	0	0.0

Table 42: Numeric-answer

Accuracy drop	Tokens saved	Token %
50.980	2997	100.0
50.980	2997	100.0
50.327	2975	99.3
46.405	2797	93.3
43.791	2575	85.9
37.908	2416	80.6
30.065	2288	76.3
25.490	2230	74.4
23.529	2162	72.1
16.340	2097	70.0
11.765	1971	65.7
4.575	1769	59.0
0.000	1097	36.6

Table 43: Search-query

Kendall τ	Tokens saved	Token %
0.827	223	100.0
0.835	215	96.7
0.881	170	76.3
0.898	145	64.9
0.908	128	57.6
0.926	113	50.8
0.942	97	43.5
0.945	88	39.7
0.956	75	33.8
0.968	60	27.1
0.979	45	20.3
0.989	19	8.7
1.000	0	0.2

H Answer Trajectory Metrics

We plot the distributions of Answer Switches, Transient Answer Flips, and the Tokens after final switch as defined in Section 2.3

H.1 Qwen3-4B

Figures 10-21 show distribution of answer trajectory metrics on Qwen3-4B.

H.1.1 MCQ

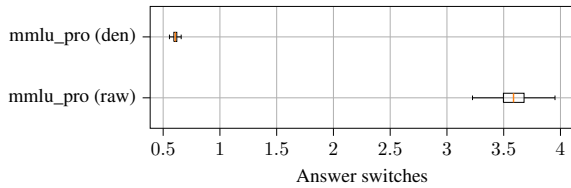


Figure 10: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-4B, evaluated on the MCQ Questions.

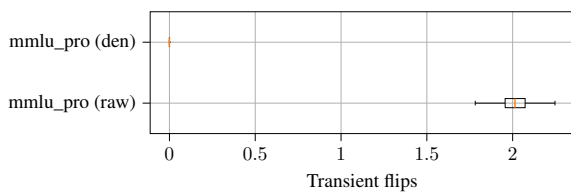


Figure 11: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-4B, evaluated on the MCQ Questions.

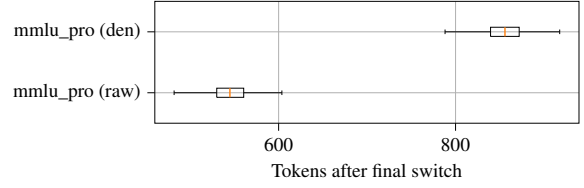


Figure 12: Distribution of the mean of tokens after final answer switch for raw and denoised outputs in Qwen3-4B, evaluated on the MCQ Questions.

H.1.2 Numeric

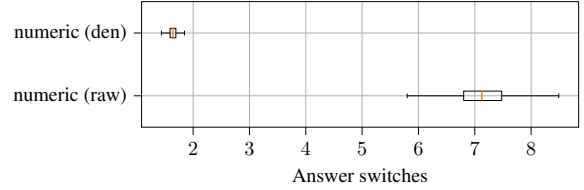


Figure 13: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-4B, evaluated on the Numeric Answers.

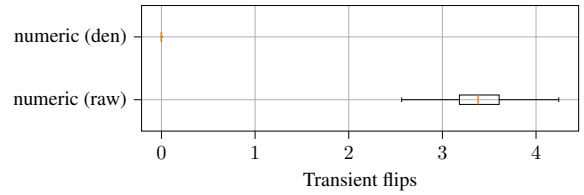


Figure 14: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-4B, evaluated on the Numeric Answers.

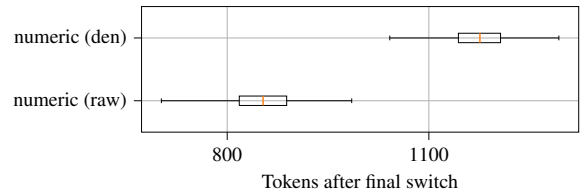


Figure 15: Distribution of the mean of tokens after final answer switch for raw and denoised outputs in Qwen3-4B, evaluated on the Numeric Answers.

H.1.3 Search

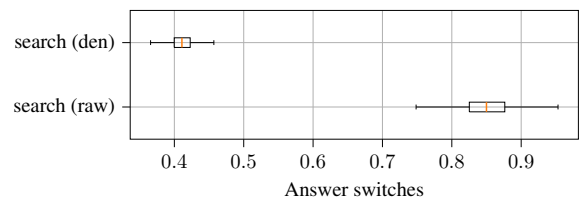


Figure 16: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-4B, evaluated on the Search Query.

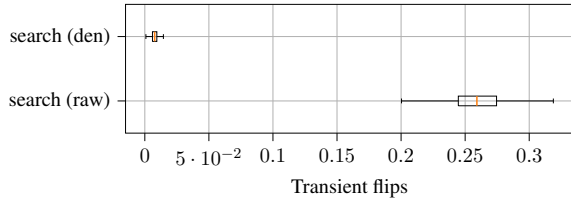


Figure 17: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-4B, evaluated on the Search Query.

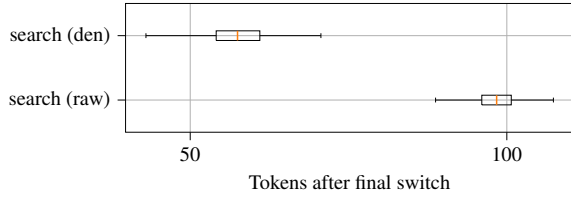


Figure 18: Distribution of the mean of tokens after final answer switch for raw and denoised outputs in Qwen3-4B, evaluated on the Search Query.

H.1.4 Tool Selection

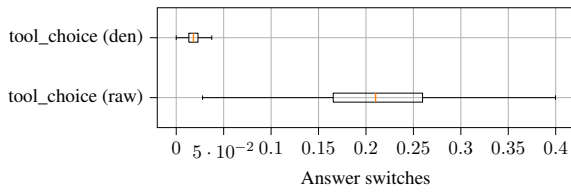


Figure 19: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-4B, evaluated on the Tool Selection.

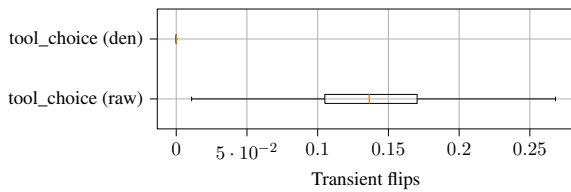


Figure 20: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-4B, evaluated on the Tool Selection.

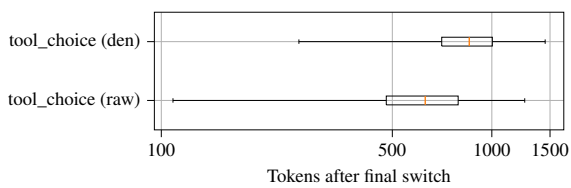


Figure 21: Distribution of the mean of tokens after final answer switch for raw and denoised outputs in Qwen3-4B, evaluated on the Tool Selection.

H.2 Qwen3-8B

Figures 22-33 show distribution of answer trajectory metrics on Qwen3-8B.

H.2.1 MCQ

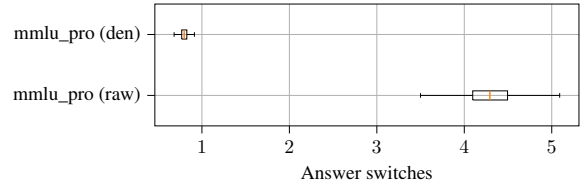


Figure 22: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-8B, evaluated on the MCQ Questions.

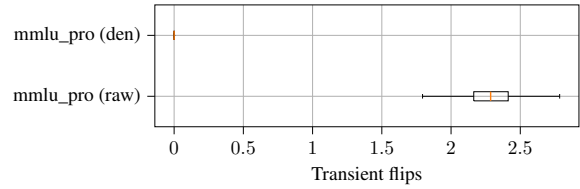


Figure 23: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-8B, evaluated on the MCQ Questions.

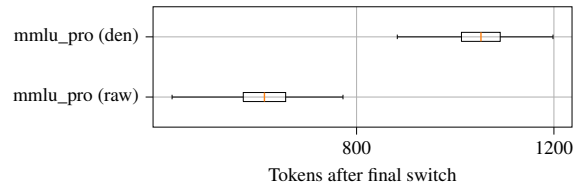


Figure 24: Distribution of the mean of tokens after final answer switch for raw and denoised outputs in Qwen3-8B, evaluated on the MCQ Questions.

H.2.2 Numeric

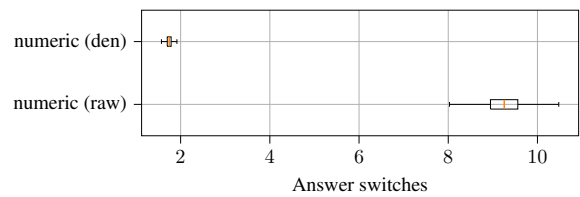


Figure 25: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-8b, evaluated on the Numeric Answers.

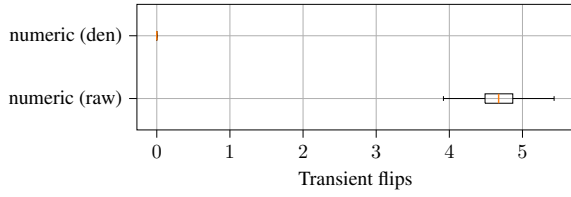


Figure 26: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-8b, evaluated on the Numeric Answers.

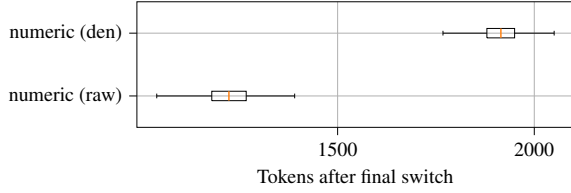


Figure 27: Distribution of the mean of tokens after final answer switch for raw and denoised outputs in Qwen3-8b, evaluated on the Numeric Answers.

H.2.3 Search

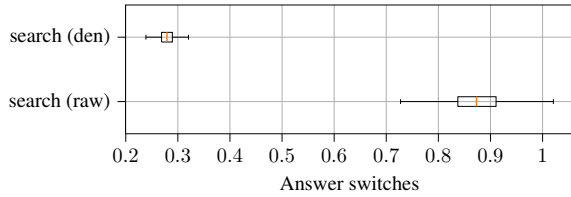


Figure 28: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-8B, evaluated on the Search Query.

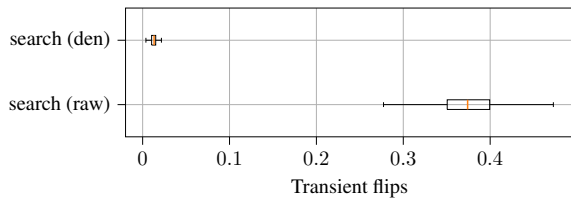


Figure 29: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-8B, evaluated on the Search Query.

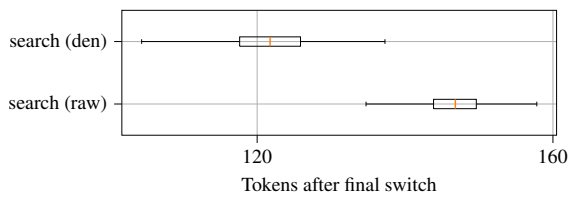


Figure 30: Distribution of tokens after final answer switch for raw and denoised outputs in Qwen3-8B, evaluated on the Search Query.

H.2.4 Tool Selection

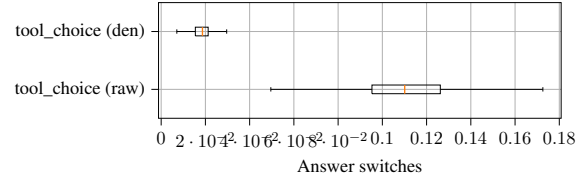


Figure 31: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-8B, evaluated on the Tool Selection.

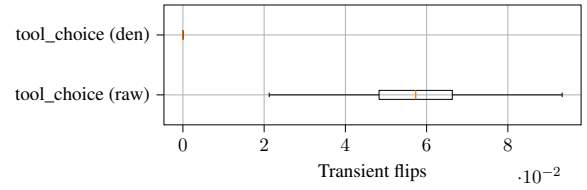


Figure 32: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-8B, evaluated on the Tool Selection.

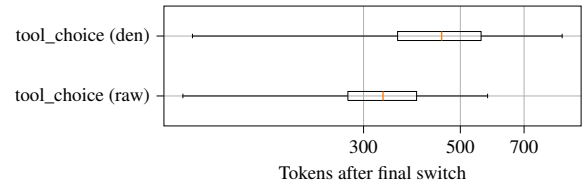


Figure 33: Distribution of the mean of tokens after final answer switch for raw and denoised outputs in Qwen3-8B, evaluated on the Tool Selection.

H.3 Qwen/Qwen3-14B

Figures 34-45 show the distribution of answer trajectory metrics on different datasets on Qwen3-14B.

H.3.1 MCQ

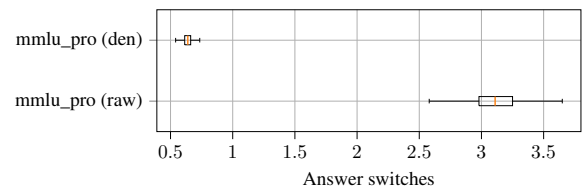


Figure 34: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-14B, evaluated on the MCQ Questions.

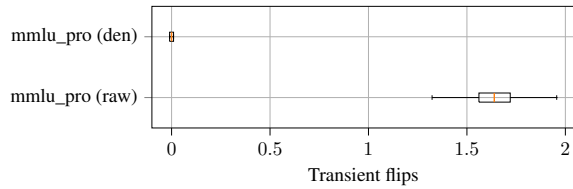


Figure 35: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-14B, evaluated on the MCQ Questions.

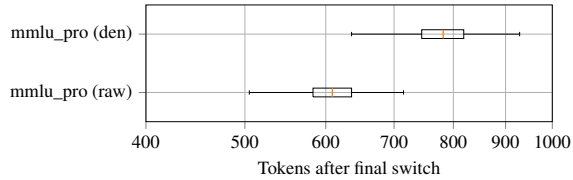


Figure 36: Distribution of the mean of tokens after final answer switch for raw and denoised outputs in Qwen3-14B, evaluated on the MCQ Questions.

H.3.2 Numeric

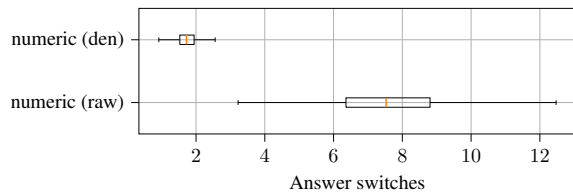


Figure 37: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-14B, evaluated on the Numeric answers.

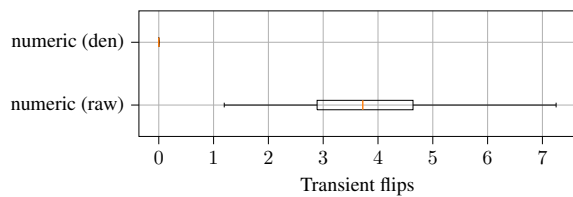


Figure 38: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-14B, evaluated on the Numeric answers.

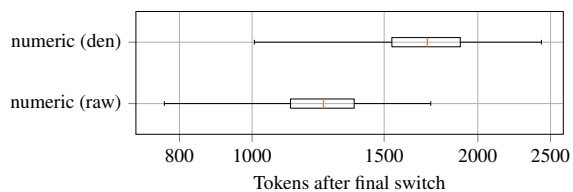


Figure 39: Distribution of the mean of tokens after final answer switch for raw and denoised outputs in Qwen3-14B, evaluated on the Numeric answers.

H.3.3 Search

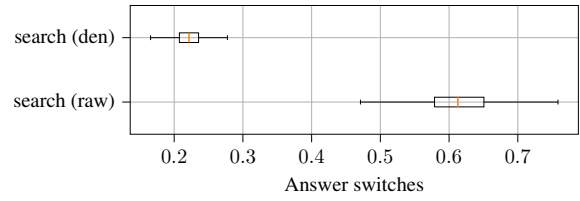


Figure 40: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-14B, evaluated on the Search query.

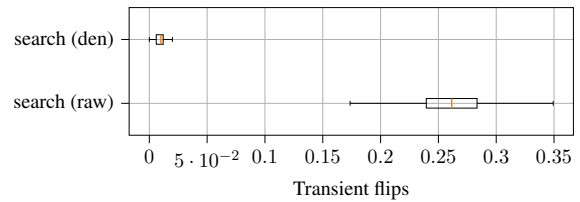


Figure 41: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-14B, evaluated on the Search query.

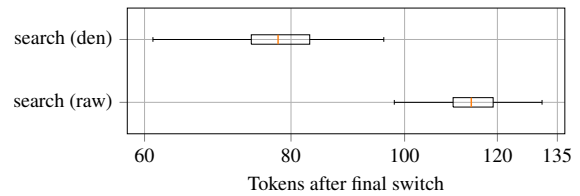


Figure 42: Distribution of the mean of tokens after final answer switch for raw and denoised outputs in Qwen3-14B, evaluated on the Search query.

H.3.4 Tool Selection

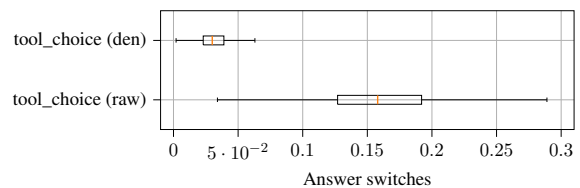


Figure 43: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-14B, evaluated on the Tool Selection.

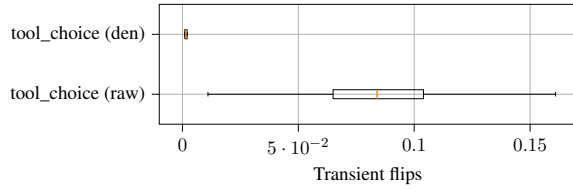


Figure 44: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-14B, evaluated on the Tool Selection.

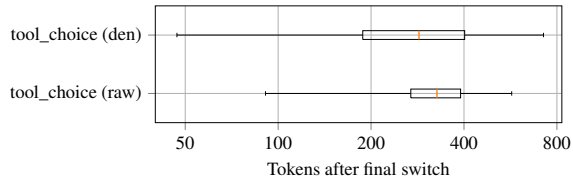


Figure 45: Distribution of the mean of tokens after final answer switch for raw and denoised outputs in Qwen3-14B, evaluated on the Tool Selection.

H.4 Qwen3-30B-A3B

Figures 46-57 show distributions of answer trajectory metrics on different datasets.

H.4.1 MCQ

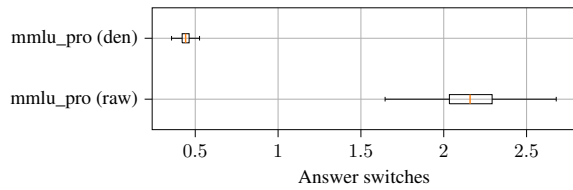


Figure 46: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-30B-A3B, evaluated on the MCQ Questions.

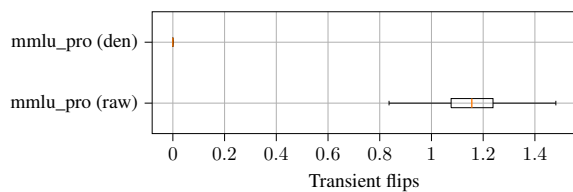


Figure 47: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-30B-A3B, evaluated on the MCQ Questions.

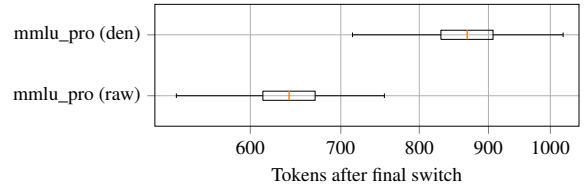


Figure 48: Distribution of the mean of tokens after final answer switch for raw and denoised outputs in Qwen3-30B-A3B, evaluated on the MCQ Questions.

H.4.2 Numeric

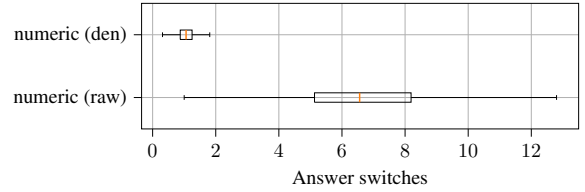


Figure 49: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-30B-A3B, evaluated on the Numeric Answers.

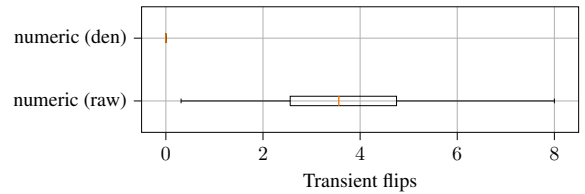


Figure 50: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-30B-A3B, evaluated on the Numeric Answers.

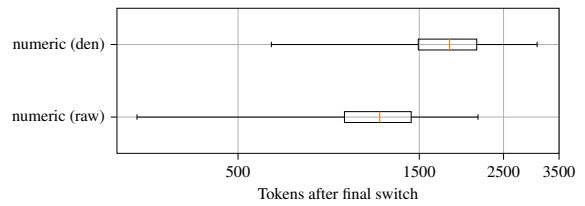


Figure 51: Distribution of the mean of tokens after final answer switch for raw and denoised outputs in Qwen3-30B-A3B, evaluated on the Numeric Answers.

H.4.3 Search

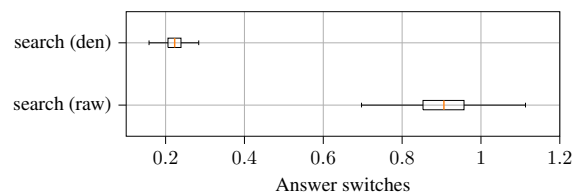


Figure 52: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-30B-A3B, evaluated on the Search Query.

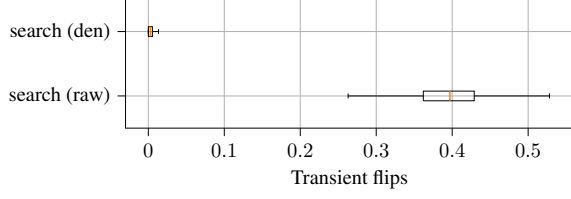


Figure 53: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-30B-A3B, evaluated on the Search Query.

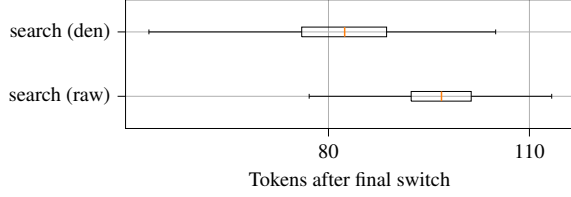


Figure 54: Distribution of the mean of tokens after final answer switch for raw and denoised outputs in Qwen3-30B-A3B, evaluated on the Search Query.

H.4.4 Tool Selection

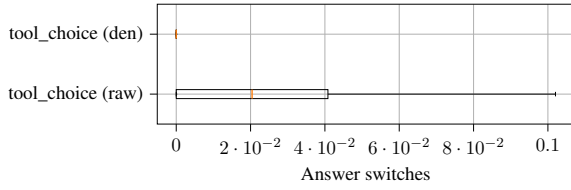


Figure 55: Distribution of the mean of answer switches for raw and denoised outputs in Qwen3-30B-A3B, evaluated on the Tool Selection.

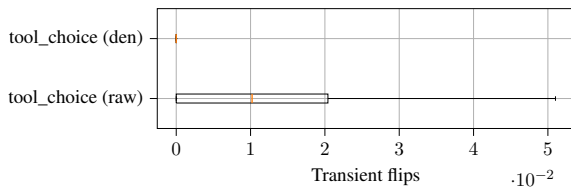


Figure 56: Distribution of the mean of transient flips for raw and denoised outputs in Qwen3-30B-A3B, evaluated on the Tool Selection.



Figure 57: Distribution of the mean of tokens after final answer switch for raw and denoised outputs in Qwen3-30B-A3B, evaluated on the Tool Selection.

I Training Details

Positive and negative training examples. Probe training examples are constructed by sampling intermediate reasoning steps from each trace rather than using all steps. Positive examples correspond to reasoning states at which computation could be halted without changing the final answer, whereas negative examples correspond to states where additional reasoning is still required.

For each example, we sample a fixed number S of reasoning steps from the trace. For a sampled step i , we extract the hidden state of the final generated token at that step from all transformer layers, denoted h_i^l for layer l . Steps satisfying $i > t^*$ are assigned a positive label, while steps satisfying $i \leq t^*$ are assigned a negative label. For examples with $t^* = -1$, all sampled steps are treated as positive. The value of S is treated as a training hyperparameter.

Because many examples exhibit no answer switch (i.e., $t^* = -1$), positive examples are more frequent than negative examples. To mitigate this imbalance, we oversample negative examples so that the two classes are approximately balanced within each minibatch.

Tool-Selection Despite the stability of tool selection, we include hidden states from tool-selection examples when training the generic probe. Although these examples provide limited signal for detecting answer switches, they expose the probe to reasoning representations from a distinct task regime, improving robustness and cross-task generalization. In particular, incorporating tool-selection data helps the generic probe learn representations associated with early decision commitment.

Optimization. We train the probe using the Adam optimizer. Learning rates are selected from the candidate set $\{1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 2 \times 10^{-3}, 5 \times 10^{-3}, 1 \times 10^{-2}, 2 \times 10^{-2}, 5 \times 10^{-2}, 1 \times 10^{-1}\}$. After fixing the best learning rate, we select the probe layer by choosing the layer whose probe achieves the highest AP on validation. The probe parameters from the best (learning rate, layer) configuration are used for all reported results for a particular model and dataset.