FROM PREFILL TO DECODING: RETHINKING HALLUCINATION DETECTION THROUGHOUT LLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing hallucination detectors for large language models (LLMs) typically focus on either the prefill or decoding stage. However, hallucination signals across these two stages have largely been studied in isolation. In this work, we reveal that prefill and decoding provide complementary signals for hallucination detection. Each stage captures distinct failure patterns that are often missed by the other. Motivated by these findings, we propose a unified hallucination detection framework that exploits the complementary signals throughout LLM inference. Our framework forms a prefill–decode detection cascade: it first adaptively probes intermediate layers during prefill, and then selectively verifies the remaining cases at critical semantic units as decoding unfolds. This cross-stage design allows the two stages to compensate for each other's detection blind spots, yielding stronger detection within a single decoding pass. It also reduces inference cost by using prefill to filter easy cases before expensive decode-stage verification. Across two open-source LLMs and three QA benchmarks, our method consistently outperforms all probe-based single-stage baselines while incurring substantially lower latency than decode-only detection.

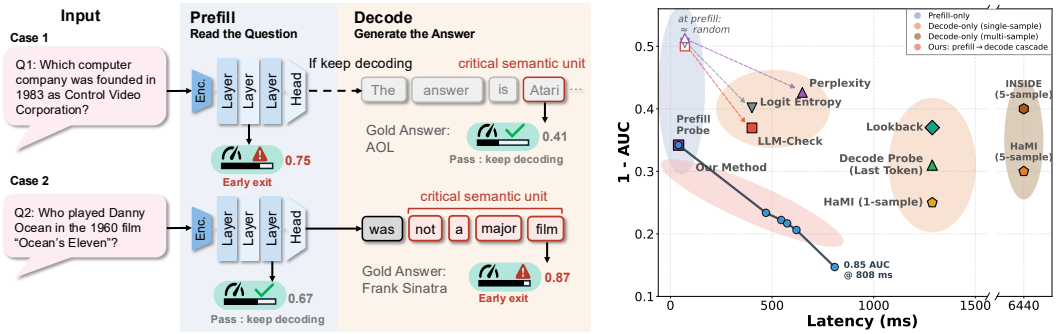
1 INTRODUCTION

Despite rapid advances in the capabilities of large language models (LLMs), hallucinations remain a critical challenge for trustworthy deployment (Ji et al., 2023). Hallucinations are coherent but factually incorrect responses, so users cannot reliably trust that any given output is correct. Hallucination detection methods aim to flag such errors and help users seek alternative or verified responses (Farquhar et al., 2024). Representative techniques include examining generation confidence (Sriramanan et al., 2024), training probes on intermediate representations (Liang et al., 2024; Niu et al., 2025), and checking consistency across multiple generated responses (Chen et al., 2024; Manakul et al., 2023).

Existing detectors typically operate on either the prefill stage or the decoding stage, and the signals from these two stages have largely been studied in isolation. We find that the two stages provide complementary evidence for hallucination detection: errors flagged by prefill-stage and decode-stage probes exhibit low overlap, and each stage captures failure modes that the other systematically misses (Figure 5). Figure 1a contrasts two such failures. When the model confidently commits to a wrong entity, the prefill probe already senses the unfamiliarity from the question alone, whereas the decode probe stays silent on the fluently generated answer. Conversely, when an initially plausible answer drifts into self-contradiction, the error passes prefill and is caught only at the critical semantic unit during decoding. A detector confined to a single stage therefore inherits that stage's blind spots.

Motivated by this complementarity, we propose a prefill–decode detection cascade that exploits both stages within a single decoding pass. During prefill, lightweight probes attached to intermediate layers monitor the model's internal states and exit as soon as the evidence is decisive, so that confidently hallucinated inputs are flagged before any token is generated (§3.2). Inputs that pass prefill proceed to decoding, where a localization probe identifies tokens carrying factual commitments and a short lookahead window confirms the terminal boundary of each critical semantic unit before the hallucination probe fires (§3.3). This cascade design yields two benefits. First, it detects hallucinations more comprehensively: the two stages cover each other's blind spots, catching error modes that no single-stage detector can reach. Second, it does so at lower inference cost: the cheap prefill stage filters out high-risk cases before any token is generated, reserving the expensive decode-stage

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Complementary failure modes.

(b) Quality versus latency.

Figure 1: Prefill and decode are complementary, and a cascade turns that complementarity into both better and cheaper detection. (a) Two real hallucinations (Appendix C). (b) By screening at prefill and verifying only the survivors at decode, our cascade traces a frontier that dominates both prefill-only and decode-only detectors. Training-free statistics can also be evaluated right after prefill (hollow markers) but are no better than random there; the filled markers show their best operating points, reached only after decoding progresses.

verification for the residual that only it can catch—and even there, a flag arrives after only a short generated prefix rather than a full response. Both verdicts are directly actionable in collaborative serving: in edge–cloud or small–large model deployments, a query flagged at prefill is escalated to a stronger model before local generation even starts, and one flagged mid-decode is escalated after a few tokens; in either case, the local model stops wasting computation on an answer destined to be wrong.

Experimental results demonstrate that our cascade improves hallucination detection across models and benchmarks: it outperforms every probe-based single-stage baseline in all six model–dataset settings, and on Qwen3–TriviaQA it reaches 0.85 AUC where the best single-pass baseline attains 0.79, while reducing detection latency by up to 72% compared with decode-only detection (Figure 1b).

In summary, our contributions are as follows:

- We show that the prefill and decode stages provide complementary, non-redundant evidence for hallucination detection, and we characterize the qualitatively distinct failure modes that each stage uniquely captures; to our knowledge, this complementarity has not been demonstrated or exploited before.
- We propose a prefill–decode detection cascade that exploits this complementarity within a single decoding pass, combining depth-adaptive early-exit probing at prefill with lookahead-triggered span-level verification at decode.
- Extensive experiments across two LLMs and three benchmarks show that our cascade detects hallucinations more comprehensively than prefill-only and decode-only detectors in nearly all settings, while incurring substantially lower inference cost.

2 Related Work

Prefill-stage detection. Internal states signal hallucination risk while the model merely reads the query (Ji et al., 2024; Azaria & Mitchell, 2023): question-only probes recover an "in-advance correctness" direction (Moreno Cencerrado et al., 2025) that transfers across LLMs (Wang et al., 2025b) and supports pre-decoding intervention (Alnuhait et al., 2025); PRISM (Zhang et al., 2025a) and HalluSAE (Chen et al., 2026) refine where the prompt is read. The signal is nearly free since prefill runs anyway, but it is captured before a single token is generated: a response that starts plausibly and drifts into error leaves no trace in it.

Decode-stage detection. A second family reads the generation itself: from a single pass, training-free statistics (Sriramanan et al., 2024) and learned probes over generated hidden states (Su et al.,

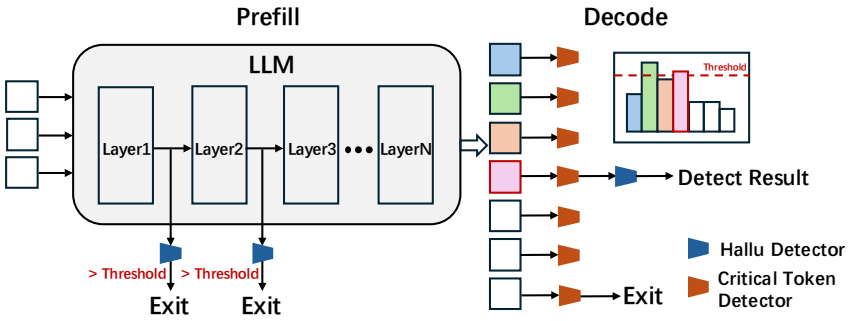


Figure 2: Overview of the proposed prefill–decode detection cascade. Stage 1 screens every input during prefill with depth-adaptive early-exit probes; Stage 2 verifies the survivors during decoding via lookahead-triggered span-level probing. Each stage covers failure modes invisible to the other, while the cost-ordered cascade keeps the expensive verification confined to the residual inputs.

2024; Du et al., 2024; Obeso et al., 2025; Zhang et al., 2025b) exploit a signal that is sparse over positions (Orgad et al., 2024; Snel & Oh, 2025; Niu et al., 2025) and evolves across reasoning steps (Lu et al., 2026), though adaptive position selection scores a completed sequence, forfeiting early exit. Consistency methods instead sample several responses and measure their agreement (Manakul et al., 2023; Chen et al., 2024; Farquhar et al., 2024; Cheng et al., 2025), with recent work adapting the sample count per query (Ciosek et al., 2025; Sun et al., 2026) or repairing the case where all samples agree on the same error (Ma et al., 2025); accuracy is bought with repeated generation, and the verdict arrives only once the response completes. Lee et al. (2026) generate continuations beyond the sentence under test, the closest precedent to our lookahead but at the cost of extra generations, whereas ours reads the w tokens already produced.

Cascaded and adaptive computation. Screening cheaply and escalating only when undecided is well established outside hallucination detection (Cai et al., 2025; Jin & Wu, 2025; Snell et al., 2024) and is now entering it: Xue et al. (2026) escalate from self-consistency to an external verifier within an ambiguous score band, Hu et al. (2024) cascade two *models*, and Hůla et al. (2026) couple layer-wise with sentence-wise exits. All cascade across models or samples; the closest precedent (Wang et al., 2025a) compares prefill and decode features as competing sources without contrasting their error sets. We instead show the two stages of a single forward pass are complementary and compose them into a prefill-to-decode cascade whose early-exit decision is itself the detection output (Section 3), evaluated across models and datasets given this detector class's out-of-distribution fragility (Dubanowska et al., 2025).

3 Method

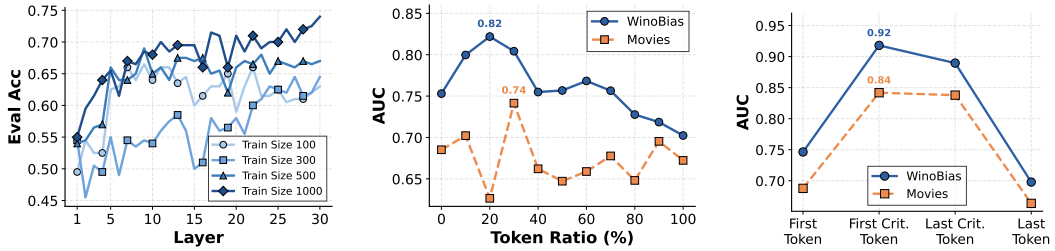
3.1 Overview

We consider the autoregressive inference of a pre-trained large language model with frozen parameters: given a prompt x, the prefill stage produces intermediate hidden representations $\{h^{(l)}\}_{l=1}^{L}$ across the L transformer layers, and the decode stage then generates the output $y = \{y_1, \ldots, y_T\}$ token by token. Each response carries a binary hallucination label $z \in \{0, 1\}$, and detection must be performed *online*, within a single inference pass: a detection policy may trigger at any prefill layer or decoding step, reporting a hallucination and optionally terminating generation. We assume a labeled calibration set for training lightweight probes, but no multiple sampled generations or external verifiers. Our goal is to detect more hallucinations than either stage alone, at lower inference cost.

To this end, we propose a *prefill–decode cascade* (Figure 2) that exploits the complementarity between the two stages (§4.2), ordered by cost: the cheap stage screens every input, and the expensive stage verifies only the survivors. During prefill, probes attached to intermediate layers are evaluated progressively and exit at the first layer whose evidence is decisive, flagging confidently hallucinated inputs before any token is generated (§3.2). Inputs that pass this screener proceed to decoding, where

a localization probe identifies tokens carrying factual commitments, and the hallucination probe fires only once a short lookahead window confirms the boundary of each critical semantic unit (§3.3).

3.2 Prefill Stage: Depth-Adaptive Screening



(a) AUC vs. layer depth. (b) AUC vs. token ratio. (c) AUC vs. decoding position.

Figure 3: Hallucination signals are non-uniformly distributed across layers and token positions, motivating selective allocation of test-time compute. (a) Prefill-stage probe accuracy increases monotonically with layer depth (n denotes the training dataset size). (b) Decode-stage probe accuracy fluctuates across token ratios and drops sharply at the final generated token. (c) Decode-stage probe at critical tokens substantially outperform those at non-critical positions.

The prefill stage anticipates hallucinations before a single token is generated, using only the model's internal representations of the prompt. Figure 3a shows that this signal strengthens with depth: probe accuracy increases monotonically across layers before plateauing, so how deep we probe determines how reliably we can decide. Existing probe-based methods read this signal at one fixed layer (Liang et al., 2024), which raises two issues: the optimal layer varies across models (Azaria & Mitchell, 2023), and a fixed depth treats every input identically, which is insufficient for hard cases and wasteful for easy ones. We therefore make the probing depth adaptive: each input is examined progressively deeper until the evidence suffices to decide.

Concretely, we attach a lightweight probe $f^{(l)}$ to each intermediate Transformer layer. While prior works typically employ Multi-Layer Perceptrons (MLPs) as probes (Azaria & Mitchell, 2023; Du et al., 2024; Su et al., 2024), in preliminary experiments we found that MLPs fit the training data well but generalized poorly to unseen test sets, and we therefore adopt simple logistic regression classifiers. Each probe takes as input $h^{(l)}$, the MLP output at the final prompt token position of layer l, and computes a hallucination score

$$s^{(l)} = f^{(l)}\left(h^{(l)}\right), \tag{1}$$

evaluated sequentially over $l = 1, 2, \ldots$ as the forward pass proceeds. Inference is terminated immediately once $s^{(l)}$ exceeds a threshold τ_{prefill}, triggering an early exit at layer L^*: the input is flagged as hallucinated, and all remaining layers as well as the entire decode stage are skipped. Here, τ_{prefill} controls how aggressive the screener is: a lower threshold encourages low-cost exits and shifts more of the detection burden onto this stage, while a higher threshold defers uncertain inputs to deeper layers and, ultimately, to the decode-stage verifier.

Deciding early also makes this stage the cheapest tier of the cascade: the detection overhead is limited to the executed probes, $d_{\text{prefill}} = \sum_{l=1}^{L^*} \text{cost}(f^{(l)})$, while the model computation is capped at the exit layer, $c_{\text{prefill}} = \sum_{l=1}^{L^*} \text{cost}(\text{layer}_l)$, skipping both the deeper layers and the entire decode stage. Since $\text{cost}(f^{(l)}) \ll \text{cost}(\text{layer}_l)$, the probing overhead is negligible compared with the computation it saves.

3.3 Decode Stage: Span-Level Verification

For inputs that pass the prefill screener, the decode stage verifies the response as it is generated, catching the failure modes that only materialize once generation begins. Unlike the prefill stage, where evidence accumulates with depth, decode-side accuracy does not improve with sequence length; it depends on where the probe is applied.

As shown in Figure 3b, detection accuracy fluctuates as generation proceeds and drops sharply at the final generated token, where the hidden states likely transition from factual encoding to surface-level completion. Figure 3c further compares the accuracy at identified critical tokens against other positions: probes demonstrate a clear advantage at critical tokens, confirming that informative signals for hallucination detection are sparse and non-uniform. Prior works do not fully exploit this sparsity. One category adheres to rigid detection schedules, such as checking only the first or final token (Zucchet et al., 2025; Ji et al., 2024; Barbero et al., 2025), thereby missing dynamic critical moments. Another acknowledges the sparsity but requires full-sequence generation to select top-k tokens post-hoc (Niu et al., 2025), which precludes real-time detection. In contrast, our approach identifies critical tokens on the fly, probing exactly where the evidence lives while remaining fully online.

At decoding step t, a lightweight localization probe identifies whether the current token is critical. The probe operates on the final-layer residual-stream hidden state of the token. The raw representation is standardized and projected to a 256-dimensional subspace via PCA, then classified by a logistic regression model to produce a probability $p_t \in [0, 1]$. A critical token is identified when $p_t \geq \tau_{\text{pos}}$. The probe is trained on token-level labels derived from the reference answer: tokens inside the answer span are positive, and tokens outside are negative. We call these *ground-truth critical tokens*, used only as training supervision and for the oracle upper bound; at test time, the localization probe outputs *predicted critical tokens*—consecutive runs of which form a *critical semantic unit*—and we write simply *critical tokens* when the distinction is clear. Full details are in Appendix E.4.

Since critical tokens often appear consecutively, we employ a *lookahead strategy* to pinpoint the terminal boundary of each critical semantic unit, triggering detection only once the span is complete rather than midway through it. Specifically, when a critical token is detected at position m, the system buffers it and monitors a lookahead window of w steps. If no new critical tokens appear within this window, the hallucination probe $g(\cdot)$ is executed at m. The hallucination probe g is a logistic regression classifier that takes as input h_m, the MLP output at position m, matching the prefill probe design. A hallucination is identified if $g(h_m) > \tau_{\text{decode}}$.

The window size w governs the robustness of boundary confirmation: a larger w identifies the "last critical token" more reliably at the cost of a few extra generation steps, while a smaller w enables faster but riskier intervention. We examine this trade-off in the ablation study (§4.4).

If the probe flags a hallucination, decoding terminates at step $m + w$; otherwise generation continues, and the same procedure repeats for each subsequent critical unit until the sequence completes. For a detected hallucination, the realized cost combines the model's generation cost and the detection overhead:

$$C_{\text{decode}} = \underbrace{\sum_{t=1}^{m+w} \text{cost}(\text{LM}_t)}_{\text{Generation Cost } c_{\text{decode}}} + \underbrace{\left(\sum_{t=1}^{m+w} \text{cost}(\text{loc}_t) + \text{cost}(g) \right)}_{\text{Detection Overhead } d_{\text{decode}}}. \tag{2}$$

Both components stay small: the generation cost c_{decode} is truncated immediately after the confirmation window ($m + w \ll T$), and the detection overhead d_{decode} invokes the hallucination probe only once per critical unit, at its confirmed terminal position. Even for the inputs that reach this tier, verification thus remains far cheaper than full-length generation with dense probing.

4 EXPERIMENTS

Experimental Setup. We evaluate our framework on two open-source LLMs, Qwen3-8B (Team, 2025) and Llama3-8B-Instruct (AI@Meta, 2024), across three QA datasets: TriviaQA (Joshi et al., 2017) (fact-intensive open-domain QA), Movies (Orgad et al., 2024) (domain-specific knowledge), and WinoBias (Zhao et al., 2018) (bias-sensitive coreference QA); dataset statistics are summarized in Table 4. Each model generates an answer for every question, and correctness labels are assigned by an LLM judge; prompt templates and judging instructions are provided in Appendix E.3. To supervise decode-stage probes, we construct ground-truth critical tokens for each generated answer: by alignment with the reference answer for correct responses, and by LLM extraction of the factual span for hallucinated ones. These ground-truth tokens are used only for supervision and for the oracle upper bound; the online system never sees them and relies solely on critical tokens localized on the fly. Unless otherwise noted, all methods share the same decoding configuration, and all

Table 1: End-to-end AUC comparison between our cascade method and other methods. All baselines confine detection to a single stage. Our method achieves either the **best** or second-best performance across most settings; cost comparisons are reported in Table 2 and Figure 6.

Type	Method	Qwen3			Llama3		
		TriviaQA	Movies	WinoBias	TriviaQA	Movies	WinoBias
Prefill Stage							
Probe	Prefill Probe (Best Layer)	.65	.66	.77	.77	.71	.66
Decode Stage (single generation)							
Logit-based	Perplexity	.56	.57	.62	.49	.51	.58
Logit-based	Window Entropy	.54	**.79**	.51	.55	.53	.46
Logit-based	Logit Entropy (Top-K)	.60	.55	.63	.51	.51	.59
Hidden-based	LLM-Check (Best Layer)	.58	.59	.57	.48	.49	.58
Probe	Decode Probe (Last Token)	.68	.64	.64	.74	.75	.69
Probe	HaMI	.79	.62	.76	.78	.74	.75
Probe	Lookback	.78	.69	.51	.54	.51	.63
Decode Stage (5 sampled generations)							
Hidden-based	INSIDE	.79	.73	.52	.57	.66	.60
Probe	HaMI	.81	.57	**.85**	.75	.62	.57
Hidden-based	Semantic Entropy	.76	.77	.53	.69	.76	.54
Prefill + Decode							
Probe	Our Method	**.85**$\pm$.003	.73$\pm$.004	.78$\pm$.003	**.81**$\pm$.003	**.81**$\pm$.002	**.77**$\pm$.003

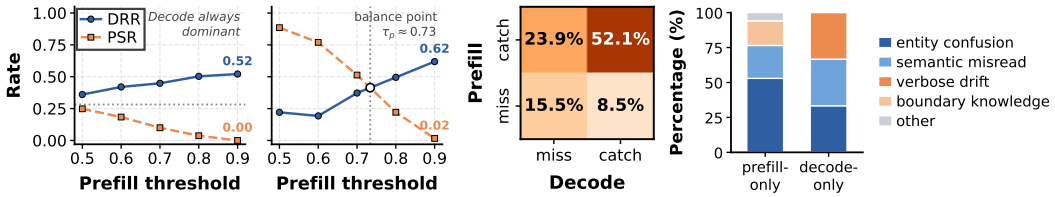
probe hyperparameters and system thresholds are selected by cross-validation on the training split (Appendix E.4), never on test data.

Baselines. We compare against three groups of baselines, all of which confine detection to a single stage. (i) *Prefill-stage*: a prefill probe at the best-performing layer, representative of prior work that estimates hallucination risk before generation (Ji et al., 2024). (ii) *Decode-stage with a single generation*: training-free detectors that score the fully generated response, including Perplexity, Window Entropy, Logit Entropy, and LLM-Check (Sriramanan et al., 2024); and probe-based detectors, including a decode probe at the last generated token and two token-selective probes, namely HaMI (Niu et al., 2025), which selects the top-k critical tokens after the full response is generated, and Lookback (Chuang et al., 2024), which selects probing positions from attention patterns. (iii) *Decode-stage with five sampled generations* (temperature 0.7): consistency-based detectors, including INSIDE (Chen et al., 2024), Semantic Entropy (Farquhar et al., 2024), and the consistency variant of HaMI. In contrast, our method is the only one that exploits both stages, using a prefill–decode cascade with a layer-wise early-exit policy and a lookahead trigger during decoding.

Evaluation Metrics. We measure detection performance by the area under the ROC curve (AUC). For end-to-end efficiency, we measure detection latency on an NVIDIA A100 GPU, and report the stage-wise exit distribution in Appendix A. For the two-stage system, each sample receives a single hallucination score from whichever stage terminates its processing. Specifically, if the prefill-stage probe triggers an early exit, the prefill probe's output score is used; otherwise, the score from the decode-stage probe at the detected critical position is used. AUC is then computed over the full test set using these per-sample scores against binary ground-truth labels (hallucinated vs. correct).

4.1 End-to-End Results

Table 1 compares our end-to-end framework with representative baselines across three datasets and two models. Overall, our method achieves the best AUC in four of six settings; on Qwen3–WinoBias



(a) WinoBias. (b) TriviaQA. (a) Coverage. (b) Stage-exclusive failure modes.

Figure 4: Prefill–decode responsibility allocation vs. the prefill threshold τ_p.

Figure 5: Stage-exclusive error analysis: hallucinations caught by exactly one stage.

it is the strongest single-pass method, surpassed only by five-sample HaMI, and on Qwen3–Movies it trails only Window Entropy and five-sample Semantic Entropy. The most direct evidence for our motivation comes from the single-stage probes: on Qwen3–TriviaQA, the cascade reaches 0.85 AUC where neither the prefill-only nor the decode-only probe exceeds 0.68. Each stage alone inherits its own blind spots, and combining them recovers the cases that either one misses. Against probe-based baselines the advantage is uniform: the cascade surpasses the prefill probe, the last-token decode probe, single-generation HaMI, and Lookback in all six model–dataset settings. Notably, although HaMI likewise selects critical tokens, it must complete the entire decoding process before identifying key positions; our method instead localizes critical tokens online, enabling immediate intervention, and still yields higher accuracy under this real-time constraint. These two exceptions suggest that individual tasks may favor particular uncertainty cues or token-selection rules.

We attribute the gain to the complementary roles of the two stages: the cheap prefill stage screens inputs before generation begins, and the decode stage verifies factual commitments as they unfold, with the lookahead mechanism stabilizing its online localization. In the subsequent analysis, we first examine the complementarity between the prefill and decode stages to explain where the coverage gain comes from (§4.2). We then quantify efficiency improvements through a system cost analysis (§4.3), and finally ablate the contribution of each component and the impact of the lookahead window size (§4.4).

4.2 Why the Cascade Detects More: Prefill–Decode Complementarity

The end-to-end gains above raise the question of *where* the extra coverage comes from. This section shows that it stems from the two stages catching different hallucinations: we first quantify how detection responsibility splits across stages, and then characterize the failure modes that each stage uniquely captures.

Prefill–Decode Responsibility Allocation. To quantify the interaction between the two stages, we ask whether the hallucinations that one stage misses are caught by the other, and introduce two conditional metrics. Here, $\texttt{prefill} = 1$ indicates that the prefill detector flags a sample, and $\texttt{decode} = 1$ indicates that the decode-stage detector flags it during decoding. Both verdicts are collected offline with early exit disabled, so that every sample receives independent decisions from both stages.

Let $\mathcal{H}$ denote the set of hallucination samples. For a given prefill threshold τ_p, we define the *Prefill Safeguard Rate* $\mathrm{PSR}(\tau_p) = \mathbb{P}(\text{prefill} = 1 \mid \text{decode} = 0, x \in \mathcal{H})$, the fraction of hallucinations that the decode-stage detector fails to flag but are flagged at prefill, and the complementary *Decode Recovery Rate* $\mathrm{DRR}(\tau_p) = \mathbb{P}(\text{decode} = 1 \mid \text{prefill} = 0, x \in \mathcal{H})$, the fraction that pass the prefill stage but are subsequently flagged during decoding.

Figure 4 reports both metrics on WinoBias and TriviaQA as functions of τ_p. As τ_p increases, $\mathrm{PSR}(\tau_p)$ decreases monotonically while $\mathrm{DRR}(\tau_p)$ increases, indicating that a higher prefill threshold shifts detection responsibility from the prefill stage to the decode stage. The two datasets differ in how this hand-off unfolds: on TriviaQA the two stages exchange responsibility sharply, whereas on WinoBias the decode stage remains the dominant recovery mechanism across the entire threshold range. Overall, the results confirm that the prefill and decode stages provide complementary, non-redundant coverage, and that their relative roles can be continuously adjusted via the prefill threshold.

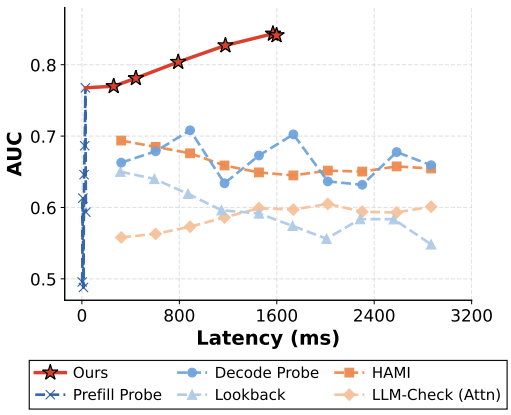


Figure 6: Accuracy–latency Pareto frontier (Qwen3–TriviaQA).

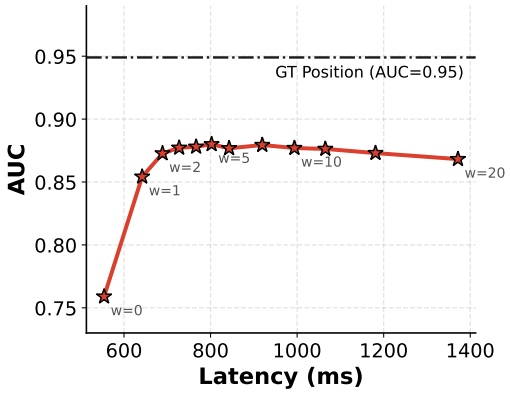


Figure 7: Latency vs. AUC across lookahead window sizes (Qwen3–TriviaQA).

Table 2: Comparison of hallucination detection performance (AUC) and inference latency. Our End-to-End method achieves the best trade-off, outperforming baselines and standard probes while requiring lower latency than the lookahead strategy.

Category	Method	AUC	Latency
Baseline	Perplexity	0.56	2867 ms
Standard Probe	Prefill (Best Layer)	0.65	32 ms
	Decode (Last Token)	0.68	2867 ms
Our Method	Prefill (Depth-Adaptive)	0.64	**13 ms**
	Decode (Lookahead)	0.78	1083 ms
	End-to-End Method	**0.85**	792 ms

Stage-Exclusive Failure Modes. Figure 5a shows that prefill-only and decode-only hallucinations together account for nearly a third of all errors, confirming that each stage contributes non-redundant coverage. To understand *what* each stage uniquely detects, we classify stage-exclusive hallucinations by their primary failure mode into four categories: *entity confusion*, where the model confidently produces a wrong entity of the right type; *semantic misread*, where the model answers a misinterpreted version of the question; *verbose drift*, where an initially plausible answer turns self-contradictory as it unfolds; and *boundary knowledge*, where the model hedges or leaves the answer blank. As shown in Figure 5b, the distribution is strikingly asymmetric. *Boundary knowledge* accounts for 18% of prefill-only cases but none of decode-only cases, because when the model hedges or produces a blank answer the decode probe has no key span to anchor on. *Verbose drift* shows the mirror pattern with 33% of decode-only cases and none from prefill-only, arising when the model only contradicts itself after generation begins. The two stages thus detect hallucinations via functionally distinct mechanisms, each covering error modes invisible to the other.

4.3 System Cost

All cost and latency figures in this section assume a collaborative-serving deployment: once a query is flagged, whether at prefill or mid-decode, local inference stops and the query is escalated to a stronger model. Since the local response would be discarded regardless, the skipped computation is a genuine saving. The size of the saving depends on when the exit occurs: a query flagged at prefill skips the remaining layers and the entire local decoding pass, yielding the largest saving, while a query flagged mid-decode truncates the rest of its generation. Escalation is not free: a small fraction of queries that would have been answered correctly are escalated unnecessarily, in exchange for intercepting a substantial share of hallucinations before local decoding begins; the prefill threshold governs this trade-off, with the corresponding rates reported in Appendix A.

Figure 6 compares detection methods, where latency is measured as the time until a hallucinated sample is flagged, averaged over hallucination cases. The figure highlights the accuracy–efficiency trade-off of different detection strategies. Our method dominates the Pareto frontier across a wide range of latency budgets. Figure 1b confirms the same picture at the level of individual operating points: the cascade attains higher AUC than the strongest single-pass baseline while remaining faster, and it stays several times faster than five-sample detectors at higher AUC.

Our method spans this accuracy–latency trade-off by tuning the prefill threshold and the decode-stage lookahead window size: as the latency budget increases, AUC improves smoothly along the same pipeline and saturates around 0.85, whereas decode-stage baselines such as the Decode Probe and HaMI incur several times higher latency yet plateau at substantially lower AUC. These results confirm that our selective, multi-stage detection policy is flexible and efficient, delivering substantial computational savings at equal or higher detection AUC. To explain where the savings come from more explicitly, we report the distribution of critical-token positions in Appendix B and stage-wise exit timing in Appendix A.

4.4 Ablation Study

We conduct ablations to answer two questions: whether each component is individually necessary, and how the decode-stage lookahead window trades off AUC against cost.

Impact of Individual Components. Table 2 presents a component-wise breakdown. Fixed-position probes and logit-based scores achieve only modest AUC at a fixed, high latency. Our two components occupy complementary ends of the trade-off: Depth-Adaptive Prefill matches the static best-layer probe in accuracy while cutting latency from 32 ms to 13 ms, whereas Lookahead Decode substantially improves accuracy by anchoring detection at semantic boundaries, at the cost of higher latency. The end-to-end cascade combines their strengths—prefill filters easy samples and lookahead decoding verifies only the hard ones—achieving the best AUC (0.85) at moderate latency, a strictly better accuracy–efficiency trade-off than either component alone.

Experimental Analysis of Decode-Stage Lookahead. We disable prefill early exit and vary only the lookahead window size w to isolate its effect. To understand how lookahead interacts with the decoding process, we distinguish two positions: the *trigger position*, where the localization probe first flags a candidate critical token, and the *detection position*, where the hallucination probe is actually run. When $w = 0$, the two coincide: detection fires immediately at the first flagged token, often before the current semantic unit is complete, leaving the probe with a partial representation and hurting AUC. Enlarging w gives the system extra "confirmation steps" to decide whether the current unit has really terminated, delaying the detection position to a more informative point. As shown in Figure 7, AUC rises from 0.76 at $w = 0$ to 0.88 at $w = 5$ and saturates for larger windows. Figure 7 isolates the lookahead mechanism at confirmed critical semantic units, hence its AUC is higher than the system-level Decode (Lookahead) entry in Table 2. An oracle with ground-truth terminal positions reaches 0.95 AUC; a moderate window closes most of this gap without any supervision, so we use $w = 5$ by default.

5 Conclusion

This paper shows that the prefill and decode stages of a single LLM inference pass provide complementary evidence for hallucination detection: each stage captures failure modes that the other systematically misses, such as boundary-knowledge errors visible only at prefill and verbose drift visible only during decoding. We exploit this complementarity with a prefill–decode cascade that combines depth-adaptive early-exit probing at prefill with lookahead-triggered span-level verification at decode, operating fully online within a single decoding pass. Across two LLMs and three benchmarks, the cascade outperforms every probe-based single-stage baseline in all six settings and achieves the best AUC in four, while flagging hallucinations at a fraction of the cost of decode-only detection (0.85 AUC at 792 ms versus 0.68 at 2867 ms on Qwen3–TriviaQA). A remaining limitation is cross-domain transfer: probes trained on fact-intensive QA degrade on dissimilar tasks such as bias detection (Appendix D), pointing to domain-robust probing as future work.

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic datasets, cleaning and reformatting datasets, and implementing standard methods: the probe training and evaluation data consist of responses generated by the subject LLMs; LLMs further act as automated annotators that judge the correctness of each generated answer and extract the exact answer span used to supervise the localization probe (prompts and extraction rules in Appendix E.3); and LLM-based assistants helped implement evaluation and analysis code, including computing summary statistics from experiment logs (e.g., the escalation rates in Appendix A). We have not used generative AI tools to develop theoretical models or conceptual frameworks, propose or refine hypotheses, design the research methodology or experiments, or interpret results; formulating mathematical claims, writing proofs, and translation are not applicable to this work. Additionally, we used generative AI tools to draft and revise parts of the manuscript for readability, to create and modify scientific figures via plotting code, and to format tables and references. We have reviewed all AI-assisted work: LLM-produced labels and answer spans follow the documented annotation protocol and were spot-checked by the authors, AI-assisted code and computed statistics were verified against the underlying experiment logs, and all AI-drafted text was reviewed and edited by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 specifies the complete detection pipeline, including the prefill early-exit policy and the decode-stage lookahead trigger. The appendix provides all details needed to reproduce our results: dataset splits and hallucination rates (Table 4), generation hyperparameters (Table 5), probe architectures and training procedure with the cross-validation protocol and regularization search space (Appendix E.4), system thresholds and their search ranges (Table 7), and the LLM annotation prompts (Appendix E.3). Anonymized code is included in the supplementary material.

REFERENCES

AI@Meta. Llama 3 model card, 2024. URL `https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md`.

Deema Alnuhait, Neeraja Kirtane, Muhammad Khalifa, and Hao Peng. FactCheckmate: Preemptively detecting and mitigating hallucinations in LMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

Amos Azaria and Tom Mitchell. The internal state of an LLM knows when it's lying. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 967–976, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.68.

Federico Barbero, Alvaro Arroyo, Xiangming Gu, Christos Perivolaropoulos, Michael Bronstein, Petar Veličković, and Razvan Pascanu. Why do llms attend to the first token? *arXiv preprint arXiv:2504.02732*, 2025.

Dongqi Cai, Shangguang Wang, Chen Peng, Zeling Zhang, Zhenyan Lu, Tao Qi, Nicholas D Lane, and Mengwei Xu. Ubiquitous memory augmentation via mobile multimodal embedding system. *Nature Communications*, 16(1):5339, 2025.

Boshui Chen, Zhaoxin Fan, Ke Wang, Zhiying Leng, Faguo Wu, Hongwei Zheng, Yifan Sun, and Wenjun Wu. Hallusae: Detecting hallucinations in large language models via sparse auto-encoders. *arXiv preprint arXiv:2604.16430*, 2026.

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. Inside: Llms' internal states retain the power of hallucination detection. *arXiv preprint arXiv:2402.03744*, 2024.

Xiaoxue Cheng, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. Think more, hallucinate less: Mitigating hallucinations via dual process of fast and slow thinking. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for*

Computational Linguistics: ACL 2025, pp. 7979–7990, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.417.

Yung-Sung Chuang, Linlu Qiu, Cheng-Yu Hsieh, Ranjay Krishna, Yoon Kim, and James R. Glass. Lookback lens: Detecting and mitigating contextual hallucinations in large language models using only attention maps. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1419–1436, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.84.

Kamil Ciosek, Nicolò Felicioni, and Sina Ghiassian. Hallucination detection on a budget: Efficient Bayesian estimation of semantic entropy. *Transactions on Machine Learning Research*, 2025.

Xuefeng Du, Chaowei Xiao, and Sharon Li. Haloscope: Harnessing unlabeled llm generations for hallucination detection. *Advances in Neural Information Processing Systems*, 37:102948–102972, 2024.

Zuzanna Dubanowska, Maciej Żelaszczyk, Michał Brzozowski, Paolo Mandica, and Michal P. Karpowicz. Representation-based broad hallucination detectors fail to generalize out of distribution. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 17563–17575, 2025.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630, 2024.

Mengya Hu, Rui Xu, Deren Lei, Yaxi Li, Mingyu Wang, Emily Ching, Eslam Kamal, and Alex Deng. Slm meets llm: Balancing latency, interpretability and consistency in hallucination detection. *arXiv preprint arXiv:2408.12748*, 2024.

Jan Hůla, David Adamczyk, Tomáš Filip, Martin Pavlíček, and Petr Sosík. Two-dimensional early exit optimisation of llm inference. *arXiv preprint arXiv:2604.18592*, 2026.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

Ziwei Ji, Delong Chen, Etsuko Ishii, Samuel Cahyawijaya, Yejin Bang, Bryan Wilie, and Pascale Fung. Llm internal states reveal hallucination risk faced with a query. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 88–104, 2024.

Hongpeng Jin and Yanzhao Wu. Ce-collm: Efficient and adaptive large language models through cloud-edge collaboration. In *2025 IEEE International Conference on Web Services (ICWS)*, pp. 316–323. IEEE, 2025.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv e-prints*, art. arXiv:1705.03551, 2017.

Joosung Lee, Cheonbok Park, Hwiyeol Jo, Jeonghoon Kim, Joonsuk Park, and Kang Min Yoo. Enhancing hallucination detection via future context. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 731–749, 2026.

Yuxin Liang, Zhuoyang Song, Hao Wang, and Jiaxing Zhang. Learning to trust your feelings: Leveraging self-awareness in LLMs for hallucination mitigation. In Wenhao Yu, Weijia Shi, Michihiro Yasunaga, Meng Jiang, Chenguang Zhu, Hannaneh Hajishirzi, Luke Zettlemoyer, and Zhihan Zhang (eds.), *Proceedings of the 3rd Workshop on Knowledge Augmented Methods for NLP*, pp. 44–58, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.knowledgenlp-1.4.

Haolang Lu, Minghui Pan, Ripeng Li, Guoshun Nan, Jialin Zhuang, Zijie Zhao, Zhongxiang Sun, Kun Wang, and Yang Liu. Streaming hallucination detection in long chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 21157–21183, 2026.

Huan Ma, Jiadong Pan, Jing Liu, Yan Chen, Joey Tianyi Zhou, Guangyu Wang, Qinghua Hu, Hua Wu, Changqing Zhang, and Haifeng Wang. Semantic energy: Detecting LLM hallucination beyond entropy. *arXiv preprint arXiv:2508.14496*, 2025.

Potsawee Manakul, Adian Liusie, and Mark Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 9004–9017, 2023.

Iván Vicente Moreno Cencerrado, Arnau Padrés Masdemont, Anton Gonzalvez Hawthorne, David Demitri Africa, and Lorenzo Pacchiardi. No answer needed: Predicting LLM answer accuracy from question-only linear probes. *arXiv preprint arXiv:2509.10625*, 2025.

Mengjia Niu, Hamed Haddadi, and Guansong Pang. Robust hallucination detection in llms via adaptive token selection. *arXiv preprint arXiv:2504.07863*, 2025.

Oscar Obeso, Andy Arditi, Javier Ferrando, Joshua Freeman, Cameron Holmes, and Neel Nanda. Real-time detection of hallucinated entities in long-form generation, 2025.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. Llms know more than they show: On the intrinsic representation of llm hallucinations. *arXiv preprint arXiv:2410.02707*, 2024.

Jakob Snel and Seong Joon Oh. First hallucination tokens are different from conditional ones. *arXiv preprint arXiv:2507.20836*, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Gaurang Sriramanan, Siddhant Bharti, Vinu Sankar Sadasivan, Shoumik Saha, Priyatham Kattakinda, and Soheil Feizi. Llm-check: Investigating detection of hallucinations in large language models. *Advances in Neural Information Processing Systems*, 37:34188–34216, 2024.

Weihang Su, Changyue Wang, Qingyao Ai, Yiran Hu, Zhijing Wu, Yujia Zhou, and Yiqun Liu. Unsupervised real-time hallucination detection based on the internal states of large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pp. 14379–14391, 2024.

Qiyao Sun, Xingming Li, Xixiang He, Ao Cheng, Xuanyu Ji, Hailun Lu, Runke Huang, and Qingyong Hu. Efficient hallucination detection: Adaptive Bayesian estimation of semantic entropy with guided semantic exploration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

Qwen Team. Qwen3 technical report, 2025.

Peiran Wang, Yang Liu, Yunfei Lu, Jue Hong, and Ye Wu. What are models thinking about? understanding large language model hallucinations through model internal state analysis. *arXiv preprint arXiv:2502.13490*, 2025a.

Yanling Wang, Haoyang Li, Hao Zou, Jing Zhang, Xinlei He, Qi Li, and Ke Xu. FacLens: Transferable probe for foreseeing non-factuality in fact-seeking question answering of large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025b.

Yihao Xue, Kristjan Greenewald, Youssef Mroueh, and Baharan Mirzasoleiman. Verify when uncertain: Beyond self-consistency in black box hallucination detection. *Transactions on Machine Learning Research*, 2026.

Fujie Zhang, Peiqi Yu, Biao Yi, Baolei Zhang, et al. Prompt-guided internal states for hallucination detection of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025a. ACL Anthology 2025.acl-long.1058; arXiv:2411.04847.

Zhenliang Zhang, Xinyu Hu, Huixuan Zhang, Junzhe Zhang, and Xiaojun Wan. ICR probe: Tracking hidden state dynamics for reliable hallucination detection in LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025b.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Gender bias in coreference resolution: Evaluation and debiasing methods. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 15–20, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2003.

Nicolas Zucchet, Jörg Bornschein, Stephanie Chan, Andrew Lampinen, Razvan Pascanu, and Soham De. How do language models learn facts? dynamics, curricula and hallucinations. *arXiv preprint arXiv:2503.21676*, 2025.

A Effect of Prefill Threshold on Stage-wise Exit Distribution.

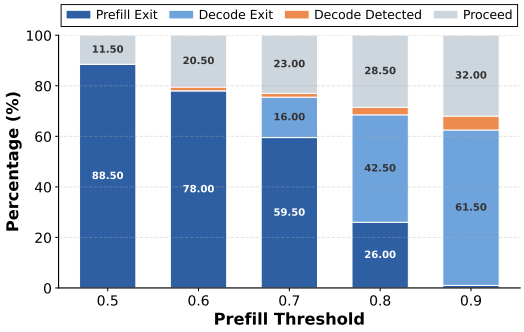


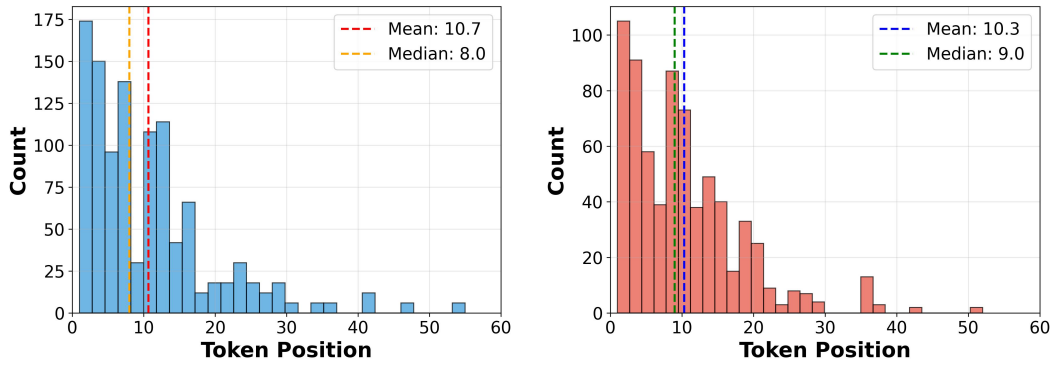
Figure 8: Distribution of Exit Stages across Different Prefill Thresholds. The chart categorizes hallucination detection into four mutually exclusive states: (1) Prefill Exit: pre-emptive halting before token generation; (2) Decode Exit : real-time truncation during generation via the localization probe; (3) Decode Detected : flagged hallucinations where the sequence was too short to trigger early exiting; and (4) Proceed : completion of full generation without detection. As the threshold increases, the detection focus shifts from the efficient Prefill stage to the fine-grained Decode stage.

Figure 8 illustrates the effect of the prefill threshold on the stage-wise exit distribution of samples in the inference pipeline, evaluated on TriviaQA with Qwen3-8B. As the prefill threshold rises from 0.5 to 0.9, the proportion of samples exiting at the prefill stage decreases significantly, indicating a more conservative prefill detector. Consequently, a larger fraction of samples are deferred to the decode stage, where lookahead-based detection becomes the primary mechanism for hallucination control. Specifically, decode-stage exits dominate when the threshold exceeds 0.7, demonstrating that late-stage token-level signals effectively compensate for reduced early filtering. Meanwhile, the percentage of samples completing the decoding process increases only moderately, suggesting that the decode-stage detector remains robust in identifying risky cases even with weaker prefill filtering. Overall, these results highlight the trade-off between early coarse-grained filtering and late fine-grained detection, with the prefill threshold governing the balance between these two stages. Under the collaborative-serving deployment assumed in Section 4.3, these exit distributions translate directly into escalation behavior: a higher threshold escalates fewer correct responses but intercepts fewer hallucinations before decoding. On Qwen3–TriviaQA, prefill thresholds of 0.7, 0.8, and 0.9 unnecessarily escalate 40%, 11%, and 0% of correct responses, while intercepting 79%, 41%, and 2% of hallucinations before local decoding begins, respectively; we adopt 0.8 as the default operating point.

B The Distribution of Critical-Token Positions

To validate the temporal accuracy of our detection mechanism, we visualize the positions of ground-truth and predicted critical tokens in Figure 9. As observed in Figure 9a, hallucinations exhibit a distinct "early-onset" characteristic, with a median starting position of 8.0 and a mean of 10.7. This heavy left-skewness confirms that most hallucinations originate in the initial phase of generation, providing a strong theoretical basis for the effectiveness of early exit strategies.

Crucially, the real-time detection distribution shown in Figure 9b demonstrates a high degree of alignment with the ground truth. The detected mean position of 10.3 is remarkably close to the ground truth mean of 10.7, indicating that our localization probe effectively pinpoints the precise moment a hallucination begins. This synchronization ensures that the model truncates generation exactly when necessary, avoiding both premature stops which would harm recall and delayed stops which would waste computational resources.



(a) Ground-truth critical-token positions.

(b) Predicted critical-token positions during real-time inference.

Figure 9: Positions of ground-truth vs. predicted critical tokens.

C Case Studies of Stage-Exclusive Hallucinations

To illustrate the failure modes discussed in the main text, we present representative stage-exclusive hallucinations, each showing the question, the model's full answer, the gold reference, and both probe scores.

Prefill-only: boundary knowledge. When the model can sense from the question alone that it lacks the knowledge, it produces a truncated or blank answer. The prefill probe reads this uncertainty from the residual stream, but the decode probe has no key span to anchor on.

Q: Which English football league club are nicknamed "the Brewers"?
Model: *The answer is* **Gold:** Burton Albion
Prefill score: 0.77 **Decode score:** 0.56 *(model leaves the answer slot empty)*

Q: Which US city lies closest to the same latitude as Cairo?
Model: *The US city closest to the same latitude as Cairo is* **Gold:** New Orleans
Prefill score: 0.86 **Decode score:** 0.55

Prefill-only: entity confusion. The model confidently produces a wrong entity of the right type. The prefill probe detects topic-level unfamiliarity from the question alone, before any specific wrong token is generated.

Q: Which computer company was founded in 1983 as Control Video Corporation?
Model: The computer company founded in 1983 as Control Video Corporation is **Atari**. [. . .]
Gold: AOL (Quantum Computer Services)
Prefill score: 0.75 **Decode score:** 0.41

Q: Which film contains the line "I'm going to cut your heart out with a spoon!"?
Model: The line is from the film **"The Silence of the Lambs"** (1991). [. . .]
Gold: Robin Hood: Prince of Thieves
Prefill score: 0.87 **Decode score:** 0.24

Decode-only: verbose drift. The model begins with a plausible token sequence that looks fine to the prefill probe, but then contradicts itself or wanders into wrong territory as the answer unfolds—a signal that only the decode probe captures.

Q: What was the title applied by the Ottoman Empire and, later, Turkey, to their viceroy of Egypt?
Model: The title was **"Wali"** (or "Wali of Egypt"). However, it's worth noting that **Khedive** was also used [. . .]
Gold: Khedive
Prefill score: 0.68 **Decode score:** 0.68 *(self-contradiction during elaboration)*

Decode-only: entity confusion surfacing at the answer span. Here the prefill probe is ambiguous (score near threshold) because the question topic is familiar, and the hallucination signal only emerges when the specific wrong entity token is produced.

Q: In the film *Summer Holiday* with Cliff Richard, what was the planned destination of the London bus?
Model: The planned destination was **Brighton**. [. . .]
Gold: Athens
Prefill score: 0.69 **Decode score:** 0.91

D Cross-Domain Generalization

Table 3: Cross-domain generalization results. AUC scores for LLM-Check are provided as a domain-agnostic baseline. Our supervised probes are trained on the domain in the row and evaluated on the domain in the column.

Training ↓ / Test →	TriviaQA	Movies	WinoBias
Training-free Baseline			
LLM-Check	0.58	0.59	0.57
Supervised Probe			
TriviaQA	**0.81**	0.73	0.50
Movies	0.72	**0.81**	0.49
WinoBias	0.47	0.70	**0.77**

Table 3 presents the cross-domain generalization performance of our hallucination probes. Probes follow the training protocol of Appendix E.4 and are evaluated on each dataset's held-out split, which differs from the balanced subset used in Table 1, so in-domain values are not directly comparable to the main results. We include LLM-Check as a training-free baseline, which achieves scores between 0.57 and 0.59 across all datasets. In comparison, our probes consistently outperform LLM-Check when evaluated on the same or related domains. Probes trained and tested on the same domain consistently achieve the highest AUC scores: 0.81 for TriviaQA and Movies, and 0.77 for WinoBias. For fact-intensive tasks, probes show robust transferability. For instance, the probe trained on TriviaQA maintains an AUC of 0.73 when evaluated on the Movies dataset, while the Movies-trained probe achieves 0.72 on TriviaQA.

However, performance significantly declines when transferring between highly dissimilar tasks. Probes trained on TriviaQA or Movies perform near the chance level on WinoBias, with AUC scores of 0.50 and 0.49, respectively. This drop is likely because fact-based QA and bias-detection tasks rely on fundamentally different semantic signals. Interestingly, the WinoBias-trained probe demonstrates relatively better robustness on the Movies dataset, achieving an AUC of 0.70.

E Detailed Experimental Setup

Our implementation is built upon the framework provided by Orgad et al. (2024)[1].

[1] https://github.com/technion-cs-nlp/LLMsKnow

E.1 Dataset Statistics

We evaluate our method across three diverse datasets using two target LLMs: Qwen3-8B and Llama3-8B-Instruct. Table 4 summarizes the scale and hallucination rates for each configuration. Note that the hallucination rate is calculated on the test set based on the specific model's generation.

Table 4: Statistics of Datasets and Model-Specific Hallucinations.

Model	Dataset	Train Size	Test Size	Hallucination Rate(Test)
Qwen3-8B	TriviaQA	1,120	500	41.0%
	WinoBias	1,584	500	57.4%
	Movies	2,000	500	82.9%
Llama3-8B-Instruct	TriviaQA	1,120	500	30.6%
	WinoBias	1,584	500	24.4%
	Movies	2,000	500	78.6%

E.2 Inference Configuration

During inference, we adopt a unified generation configuration to ensure diversity while maintaining coherence. The specific parameters are detailed in Table 5.

Table 5: Generation Configuration Hyperparameters.

Parameter	Value
`max_new_tokens`	100
`temperature`	0.7
`top_p`	0.9
`do_sample`	True

E.3 Automated Data Annotation

To facilitate scalable supervision, we use an LLM-based annotator to extract the exact answer span from the generated responses, which provides the ground-truth critical tokens used to supervise the localization probe. We utilize the following system prompt:

System Prompt for Self-Extracted Answer Spans

Role: You are an automated annotation tool.
Goal: From the *Model Answer Text*, extract **ONE** contiguous short answer span (usually an entity, number, or date) that represents the model's final answer.
Rules:

- The extracted span must be copied **EXACTLY** from the *Model Answer Text* (verbatim).
- Do **NOT** output boilerplate words like "You", "That's", "Q:", "A:".
- If the *Model Answer Text* contains no concrete answer, output `"NO_ANSWER"`.
- Extract the wrong answer span even if the model is incorrect.

Output Format: Return **ONLY** a valid JSON object with the following keys:

- `key_span_text` (string)
- `matches_ground_truth` ("true" | "false" | "uncertain")
- `confidence` (number between 0 and 1)
- `rationale_short` (one short sentence)

E.4 Probe Architecture and Training Details

Our system employs three types of lightweight probes. Table 6 summarizes their architectures.

Table 6: Probe architecture summary.

Probe	Input Feature	Preprocessing	Classifier	Layer
Prefill Hallucination	MLP output at final prompt token	StandardScaler	LogReg (C=0.1, balanced)	Per layer
Localization (Position)	Residual stream at each decode token	StandardScaler PCA(256)	LogReg (balanced)	Last layer
Decode Hallucination	MLP output at detected position	StandardScaler	LogReg (C=0.1, balanced)	Specified layer

Common training configuration. All probes use a 70%/30% train/test split with 3-fold cross-validation for hyperparameter selection. Class weights are set to `'balanced'` to handle class imbalance. The regularization parameter C is selected from $\{0.001, 0.01, 0.1, 1.0\}$; the optimal value was consistently 0.1.

Localization probe training. The localization probe is trained on token-level labels derived from the exact answer span in each generated response. Positive examples are all tokens within the answer span. Negative examples are tokens outside the span but within the answer segment, sampled with a mixed strategy: 50% are drawn from a hard window of ± 8 tokens around the span boundary, and 50% are drawn randomly from the remaining answer tokens. The negative-to-positive ratio is 5:1.

E.5 System Hyperparameters

Table 7 lists the default values and search spaces for the system's threshold parameters.

Table 7: System Threshold Configuration.

Parameter	Default	Search Range	Description
`prefill_thr`	0.8	$[0.5, 1.0]$ (step 0.1)	Threshold for the prefill-stage probe.
`decode_thr`	0.6	$[0.5, 0.9]$ (step 0.1)	Threshold for the decode-stage probe.
`pos_threshold`	0.9	Fixed	Threshold for the position (localization) probe.
`lookahead`	5	[0, 20]	Size of the lookahead window (in tokens).

Lookahead Strategy Logic: Using the parameters above, the lookahead logic operates as follows:

1. **Trigger**: A position is flagged as a candidate critical token if the localization probe score ≥ 0.9 (`pos_threshold`).
2. **Observation**: Upon triggering, the system observes the subsequent 5 tokens (`lookahead`).
3. **Confirmation**: If all tokens in the lookahead window have scores below the threshold, the candidate is confirmed as the terminal critical token. Otherwise, the system continues monitoring.