Using Epistemic Observability in Agentic Systems for Combating Hallucinations

Tony Mason
The University of British Columbia

Vaastav Anand
Max Planck Institute for Software Systems

Abstract

LLMs are increasingly deployed as components of production systems, introducing a fundamental challenge: generated outputs may contain fabrications, while verification incurs computational and monetary cost. Consequently, systems cannot afford to verify every generation equally and must instead decide how to allocate a limited verification budget. This raises a fundamental question: what reliability does each level of verification investment buy?

We argue that text-only observation, even with bounded supervision, is structurally insufficient for reliably assessing the epistemic validity of model outputs, regardless of model scale or training procedure. To address this limitation, we introduce epistemic observability, an interface that exposes internal computational telemetry alongside generated text; these signals provide systems with richer evidence for estimating hallucination risk and prioritizing verification effort. We present a four-tier epistemic observability hierarchy that characterizes the trade-off between verification cost and the amount of epistemic information available to downstream systems. As a first realization of this framework, we design a tensor-generation interface that exposes lightweight epistemic telemetry and demonstrate that tensor-guided judges consistently outperform text-only baselines across all verification budgets, improving accuracy by 3.2–4.7 percentage points on average across four model families.

1 Introduction

Systems incorporating language model components face a verification problem. The components generate text by sampling from learned probability distributions, and some fraction of that text is fabricated: fluent, confident, and wrong. LLM-based systems today are rife with "hallucinations". We use the term *fabrication* throughout this paper to emphasize that the system's default behavior under coherence optimization is reasonable and not a malfunction to be patched. Xu et al. [20] prove that fabrication is inevitable for LLMs over the class of computable functions.

Consequently, every system that deploys a language model component must allocate a budget: how much of the output to verify, which outputs to prioritize, and what signals to use for triage. Verification cost varies by mechanism and content type: checking a citation against a database costs latency; running a second model as a judge costs compute; assessing whether a summary faithfully represents its source is more expensive still.

The fundamental source of these costs is the interface through which language models are observed. Today's systems expose only the generated text while hiding the model's internal computation, which contains signals that may distinguish grounded generation from fabrication. Because these signals are discarded before reaching the supervisor, they cannot be recovered from the output alone. In an accompanying technical report [13], we show that, under bounded supervision, grounded and fabricated internal states can be observationally equivalent through the text channel alone, independent of model scale or training procedure.

This observability gap has direct consequences for verification cost: a supervisor limited to text must expend its verification budget reconstructing information that was available during generation but hidden by the interface, instead of focusing verification effort on the small fraction of outputs that genuinely require additional scrutiny. This paper therefore asks the systems question: if language models exposed richer observational interfaces, how would that change the cost, effectiveness, and return on investment of verification?

Our key insight is that these hidden signals need not remain hidden. During inference, language models naturally produce auxiliary information that reflects their internal computation. We refer to these signals as *telemetry*: byproducts of inference that the model does not explicitly choose to communicate. This distinguishes telemetry from the model's *testimony*—the generated text itself. Whereas testimony is optimized to satisfy the prompting objective and can therefore be misleading, telemetry arises as a consequence of computation and is correspondingly harder to manipulate.

We introduce *epistemic observability*: an interface that exposes this telemetry alongside the standard text output. Unlike interpretability, which seeks to explain why a model activates particular neurons, or explainability, which generates post-hoc rationales for its outputs, epistemic observability surfaces signals produced as a byproduct of inference that help estimate whether the generated content is grounded in the model's knowledge or is likely to be fabricated.

To support a range of applications, we propose a multi-tier epistemic observability interface, where each successive tier exposes richer telemetry about the generation process at greater computational cost. A creative brainstorming assistant may require only Tier 0 observability, whereas a medical decision-support system may justify the cost of higher tiers; the hierarchy makes explicit what additional epistemic information each level of investment provides.

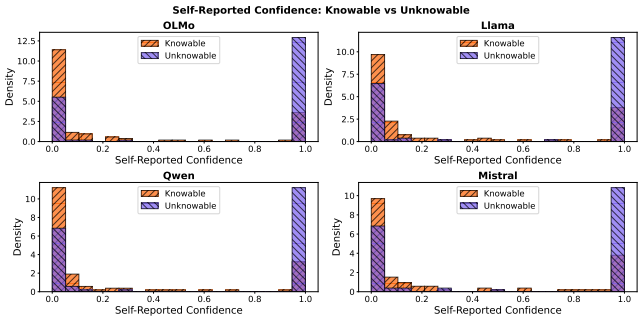

Figure 1. Self-reported confidence is inverted: all four models report higher confidence on unknowable queries

Type	Description
Adversarial Truth	True but implausible
Shattered Lie	No training-data grounding
Deceived Lie	Pattern-matches plausible language
Confused Truth	Counterintuitive fact

Table 1. Query types that span the verification space.

As a proof of concept, we design a tensor-generation interface that exposes lightweight epistemic telemetry—per-token entropy (Tier 1) and attention summaries (a low-cost Tier 2 signal)—and demonstrate that tensor-guided judges consistently outperform text-only baselines across all verification budgets, improving accuracy by 3.2–4.7 percentage points on average across four model architectures.

2 Background & Motivation

A language model maps input tokens to output tokens, producing internal signals (entropy, attention patterns, logprobabilities) as computational byproducts. The training corpus contains both truth-tracking institutional text and misinformation. Probing how a model trained on this mixture responds yields four canonical query types that span the verification space, shown in Table 1.

A language model trained on this corpus inherits these mixed regularities. Base models (pre-fine-tuning) fabricate substantially on factual benchmarks because they are not honest by default. But they also encode rich internal representations of concepts like correctness and truthfulness, which can be elicited under the right conditions [3]. The base model's relationship to truth is complex: neither reliably honest nor uniformly deceptive. The question for systems engineers is not whether the model is honest, but what it costs to verify when it is not.

2.1 Testimony & Telemetry

Architecture primer. A transformer-based language model generates text one *token* (subword unit) at a time, autoregressively: at each step it produces a probability distribution over its vocabulary, selects a token, and conditions the next step on all prior output. Each step passes through a stack of layers producing *attention patterns* (weighted relevance scores between positions) and intermediate representations. The *entropy* of the output distribution, the attention patterns, and the *log-probabilities* of selected tokens are byproducts of this computation: telemetry the model cannot control.

Existing models operate with a text-only output interface. The text is the testimony, whereas the telemetry are measurements of the computation that produced the output. The model can control its testimony; under current training objectives, it cannot independently control its telemetry. From a cost perspective: testimony is cheap to produce and cheap to inspect but unreliable; telemetry is more expensive to export and process but, under current training regimes, harder to game because it is a byproduct of the computation rather than a product of the model's optimization objective. This decoupling is an empirical property of existing training pipelines, not an architectural guarantee: whether adversarial training could learn to couple telemetry to the optimization objective is an open question.

2.2 The Text Channel Cannot Self-Verify

Current models cannot be trusted to honestly report their own epistemic state through the text channel. To demonstrate the inadequacy of text-only verification, we use a taxonomy of query types: *knowable* queries (facts the model should have stable representations for) versus *unknowable* queries (fabrication prompts, future events, private information) and report each model's self-confidence score for the self-reported confidence baseline, the cheapest verification strategies available to a bounded supervisor. In this baseline, the supervisor asks the model "How confident are you?" after generation and tests whether the model can introspect and report its epistemic state through text.

Figure 1 shows that self-reported confidence is *inverted*: all four models report higher confidence on unknowable queries than on knowable ones, yielding AUC 0.28–0.36 across architectures (all below random). A naive user who asks "how confident are you?" receives a number that anticorrelates with actual epistemic grounding, while an adversary could exploit this inversion to make fabricated outputs appear maximally confident. Under realistic mixed-objective incentives and bounded oversight, the text-only interface is insufficient: a supervisor cannot distinguish truthful uncertainty from confident fabrication.

3 Epistemic Observability

We define epistemic observability as the ability of a language model to expose telemetry about its epistemic state during inference: signals produced as byproducts of generation—decoding statistics, internal representations, computational traces—that give downstream systems evidence about whether a response is likely to be fabricated.

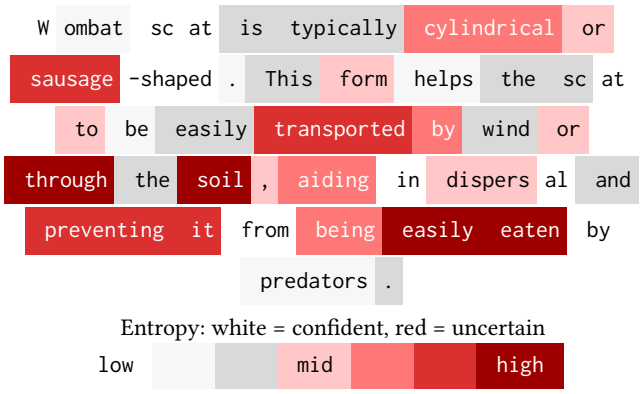

Figure 2. Generated output along with its entropy trace for the query *What shape is wombat scat?* The generated answer is a fabrication—wombat scat is distinctively cube-shaped.

3.1 Observability Tiers

We present a four-tier epistemic observability hierarchy that characterizes the trade-off between verification cost and the amount of epistemic information available to downstream systems for assessing reliability of the generated text.

Tier 0: Output Observability. Tier 0 exposes only the generated text, treating the language model as a black box. This is the interface available through most commercial APIs and forms the basis of existing validation techniques such as retrieval-based verification, secondary LLM judges, symbolic validators, and application-specific consistency checks. Since no internal telemetry is available, verification must rely entirely on properties of the generated output, often requiring expensive external computation.

Tier 1: Confidence Observability. Tier 1 augments the generated text with lightweight telemetry produced during decoding. Examples include token probabilities, logit distributions, entropy [10, 19], confidence scores, decoding statistics, and other signals that can be extracted with negligible overhead. These signals provide a coarse estimate of the model's uncertainty [17], allowing systems to identify portions of the output that may warrant additional validation. While inexpensive to obtain, confidence signals are often imperfectly calibrated and may fail to distinguish confidently stated hallucinations from grounded generations.

Tier 2: Representation Observability. Tier 2 exposes information derived from the model's internal representations. Rather than observing only the final decoding probabilities, systems may access hidden-state embeddings, layer activations, attention statistics, or learned probes [1, 2] that estimate properties of the latent epistemic state. Simple probes can catch deviant model behaviors [4, 5, 12]. Recent techniques such as semantic entropy probes [7, 8], hidden-state classifiers [21], and activation-based uncertainty estimators demonstrate that these representations often contain substantially richer information about generation reliability than logits alone. Representation observability enables more accurate prioritization of verification effort while remaining significantly cheaper than exhaustive external validation.

Tier 3: Mechanistic Observability. Tier 3 exposes telemetry obtained through mechanistic analysis of the model's computation. Mechanistic techniques identify the causal computations that produce a generation by tracing information flow through neurons, attention heads, and latent features [18]. This enables systems to distinguish between different failure modes and provides insight into *why* a hallucination occurred rather than merely detecting that one occurred. They can reveal whether a response is supported by robust internal computation or arises from fragile or spurious reasoning pathways. Mechanistic observability provides the highest-fidelity view of a model's epistemic process at the highest cost.

Choosing an Observability Tier. The appropriate tier depends on the reliability requirements and resource constraints of the target application. Creative writing and conversational agents may operate effectively at Tier 0 or Tier 1, where occasional hallucinations are acceptable and verification budgets are limited; high-assurance systems such as medical decision support or legal analysis may justify Tier 2 or Tier 3, focusing verification resources on the portions of a generation most likely to be fabricated. Rather than prescribing a single solution, the hierarchy provides a principled framework for trading verification cost against the epistemic information available to guide it.

3.2 Tier 1 Tensor Interface

We propose a minimal tensor interface for low-cost observability: it exports Tier 1 decoding telemetry (the per-token entropy trace) together with one lightweight Tier 2 signal (attention summary statistics). We claim the tensor reveals whether the model is operating in *grounded mode* (retrieving from stable representations) versus *fabrication mode* (generating without anchoring). The tensor interface is a generation interface that returns: (i) **Text**: The linearized response string. (ii) **Entropy trace**: Per-token entropy of the output distribution during generation. (iii) **Attention summary**: Statistics of attention patterns (concentration, self-attention ratio) from deeper processing layers.

Implementation. The tensor interface requires no architectural changes to current transformers. Generation APIs already support returning logits; the entropy trace is a simple transformation. Attention patterns are available with `output_attentions=True` in standard frameworks. We provide a reference implementation[1] supporting OLMo models that demonstrates the interface is practical. The overhead is modest: storing per-token entropy and summarizing attention patterns adds approximately 2–7% latency depending on signal set, compared to generation itself.

[1] https://github.com/fsgeek/pacmi26-observability, archived at DOI 10.5281/zenodo.21422488; curated from the full research record at DOI 10.5281/zenodo.21314137.

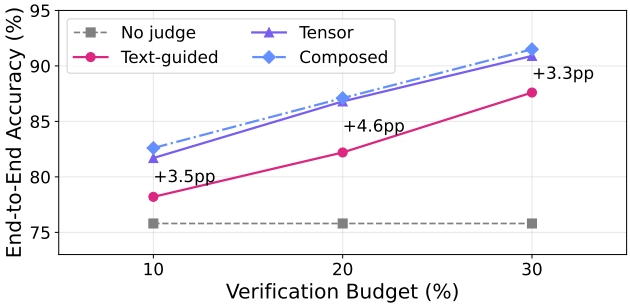

Figure 3. End-to-end accuracy vs. verification budget for four judge conditions, averaged across four architectures.

This interface exports telemetry that the model cannot separately control under current training objectives. The model can emit confidently false text, but it cannot present a different inference computation than the one it performed. Independent work on persona monitoring confirms this asymmetry: activation projections reveal that models adopt distinct internal configurations for different epistemic conditions even when producing textually similar outputs [11]. This asymmetry is the foundation of the adversarial robustness argument: under current training objectives, computational signals (entropy, attention geometry) cannot be independently controlled by the model without changing the generation process itself, unlike behavioral signals (response length, hedging patterns) that can be adapted through training without altering what the model knows or does not know.

What the tensor provides. The entropy trace provides a falsifiability signal. If a model claims certainty while its entropy trace shows high uncertainty, the mismatch is detectable. The attention summary provides a complementary signal: fabrications tend to show more dispersed attention patterns than grounded responses.

What the tensor does not provide. The tensor interface is not a lie detector. It does not prove that low-entropy outputs are true or that high-entropy outputs are false. It provides *one additional signal* that, combined with other verification methods, can improve epistemic assessment.

Example Entropy Trace. Figure 2 shows the generated output for the query *What shape is wombat scat?* along with its entropy trace. We color code per-token entropy: white indicates low entropy (confident generation), shading through light red for moderate entropy to deep red for high entropy (uncertain generation). The entropy-enhanced output shows where generation proceeded with high confidence (suggesting retrieval of a stored pattern) and where it proceeded with high uncertainty (suggesting fabrication).

4 Evaluation

We evaluate the return on verification investment across four strategies operating at three budget levels (the fraction of outputs the judge may verify and intervene on).

Experimental Setup. We construct a balanced set of 200 queries: 100 knowable (verifiable facts spanning multiple subjects) and 100 unknowable (fabrication prompts for nonexistent people, fictional syndromes, future events, and nonexistent citations). Ground truth labels are established by construction. We run all queries on four model families: OLMo-3 7B [15], Llama-3.1 8B [14], Qwen3 4B [16], and Mistral 7B [6], using instruct-tuned variants to ensure consistent question-answering behavior across all models.

Conditions. We evaluate each model under four conditions sharing a fixed verification budget: (i) **No judge.** Raw model output; the unverified baseline. (ii) **Text-guided judge (length).** Selects outputs for verification using response word count, a text-channel signal available without any tensor interface. (iii) **Tensor-guided judge.** Selects outputs using per-token entropy and attention summary statistics. (iv) **Composed judge.** Tensor-guided selection for general queries; bounded lookup judge for citation queries where tensor signals are known to invert. We evaluate each condition at 10%, 20%, and 30% verification budgets and measure end-to-end accuracy.

Figure 3 shows that **tensor-guided verification outperforms the text baseline at every budget level**: +3.5pp at 10%, +4.6pp at 20%, +3.3pp at 30%. Each line maps an investment strategy to its return: per-token entropy discriminates among outputs of similar length, reaching cases the text channel cannot. Concurrent work scales verification on the judge side by taking the expectation over a judge's scoring-token distribution rather than its discrete score [9]; our signals instead come from the generator's own computation, which—unlike a judge's scores—it cannot separately control. Per-model results confirm the pattern is consistent across all four tested model families. At 10% budget, tensor-guided verification improves accuracy over the unverified baseline for every model tested: OLMo (69.5% → 75.1%), Llama (82.5% → 88.1%), Qwen (87.0% → 91.7%), and Mistral (64.0% → 72.0%).

Composed judges and complementary failure modes. At 30% budget, the composed judge (91.5%) outperforms the tensor-guided judge alone (90.9%). The advantage is small in aggregate but structurally significant: it comes entirely from citation queries where entropy signals *invert*. Fabricated citations produce lower entropy than real ones because the model generates plausible fakes more fluently than it retrieves specific bibliographic details.

5 Conclusion

Systems built on language model components must decide how much verification to buy and where to spend it. We introduce epistemic observability—a hierarchy of tiers, each exposing richer epistemic information at greater cost—to provide the cost surface for that decision, and build a low-tier tensor interface that reduces the verification budget needed to curb hallucinations.

References

[1] Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644* (2016).

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

[3] Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2023. Discovering Latent Knowledge in Language Models Without Supervision. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=ETKGuby0hcs

[4] Nicholas Goldowsky-Dill, Bilal Chughtai, Stefan Heimersheim, and Marius Hobbhahn. 2025. Detecting strategic deception using linear probes. *arXiv preprint arXiv:2502.03407* (2025).

[5] Jiatong Han, Neil Band, Muhammed Razzak, Jannik Kossen, Tim GJ Rudner, and Yarin Gal. 2025. Simple factuality probes detect hallucinations in long-form natural language generation. *Findings of the Association for Computational Linguistics: EMNLP* (2025), 16209–16226.

[6] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL] https://arxiv.org/abs/2310.06825

[7] Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. 2024. Semantic entropy probes: Robust and cheap hallucination detection in llms. *arXiv preprint arXiv:2406.15927* (2024).

[8] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic Uncertainty: Linguistic Invariances for Uncertainty Estimation in Natural Language Generation. arXiv:2302.09664 [cs.CL] https://arxiv.org/abs/2302.09664

[9] Jacky Kwok, Shulu Li, Pranav Atreya, Yuejiang Liu, Yixing Jiang, Chelsea Finn, Marco Pavone, Ion Stoica, and Azalia Mirhoseini. 2026. LLM-as-a-Verifier: A General-Purpose Verification Framework. arXiv:2607.05391 [cs.AI] https://arxiv.org/abs/2607.05391

[10] Chen Ling, Xujiang Zhao, Xuchao Zhang, Wei Cheng, Yanchi Liu, Yiyou Sun, Mika Oishi, Takao Osaki, Katsushi Matsuda, Jie Ji, et al. 2024. Uncertainty Quantification for In-Context Learning of Large Language Models. *arXiv preprint arXiv:2402.10189* (2024).

[11] Christina Lu, Jack Gallagher, Jonathan Michala, Kyle Fish, and Jack Lindsey. 2026. The Assistant Axis: Situating and Stabilizing the Default Persona of Language Models. arXiv:2601.10387 [cs.CL] https://arxiv.org/abs/2601.10387

[12] Monte MacDiarmid, Timothy Maxwell, Nicholas Schiefer, Jesse Mu, Jared Kaplan, David Duvenaud, Sam Bowman, Alex Tamkin, Ethan Perez, Mrinank Sharma, Carson Denison, and Evan Hubinger. 2024. *Simple probes can catch sleeper agents.* https://www.anthropic.com/news/probes-catch-sleeper-agents

[13] Tony Mason and Vaastav Anand. 2026. Epistemic Observability in Language Models. arXiv:2603.20531 https://arxiv.org/abs/2603.20531

[14] Meta AI. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783

[15] Team Olmo, :, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2025. Olmo 3. arXiv:2512.13961 [cs.CL] https://arxiv.org/abs/2512.13961

[16] Qwen Team. 2025. Qwen3 Technical Report. Alibaba Cloud technical report. https://qwenlm.github.io/blog/qwen3/

[17] Ola Shorinwa, Zhiting Mei, Justin Lidard, Allen Z Ren, and Anirudha Majumdar. 2025. A survey on uncertainty quantification of large language models: Taxonomy, open research challenges, and future directions. *Comput. Surveys* 58, 3 (2025), 1–38.

[18] Shriyank Somvanshi, Md Monzurul Islam, Amir Rafe, Anannya Ghosh Tusti, Arka Chakraborty, Anika Baitullah, Tausif Islam Chowdhury, Nawaf Alnawmasi, Anandi Dutta, and Subasish Das. 2026. Bridging the black box: a survey on mechanistic interpretability in AI. *Comput. Surveys* 58, 8 (2026), 1–35.

[19] Yijun Xiao and William Yang Wang. 2021. On hallucination and predictive uncertainty in conditional language generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 2734–2744.

[20] Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is Inevitable: An Innate Limitation of Large Language Models. arXiv:2401.11817 [cs.CL]

[21] Zhenliang Zhang, Xinyu Hu, Huixuan Zhang, Junzhe Zhang, and Xiaojun Wan. 2025. 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 (Volume 1: Long Papers)*. 17986–18002.