The Sakshi Witness Loop:

An Entropy-Guided Architecture for Self-Monitoring in Language Models

A Proof-of-Concept Study

Nirmal G. Parmar

Independent Researcher
Surat, Gujarat, India

December 2025

Preprint submitted to arXiv and Zenodo

Abstract

Large Language Models (LLMs) operate as feed-forward probabilistic systems that lack introspective access to their own epistemic uncertainty during inference. This architectural limitation frequently results in "hallucinations"—confident assertions of factually incorrect information. We introduce the **Sakshi Witness Loop**, a novel monitoring architecture that tracks the Shannon entropy of a model's output distribution during token generation. When entropy exceeds a calibrated threshold, indicating high uncertainty, a secondary "Witness" process can trigger interventions such as epistemic acknowledgment or reasoning augmentation.

Drawing conceptual inspiration from Advaita Vedanta's notion of *Sakshi* (witness consciousness), we implement a computational observer that monitors cognitive stability without altering model weights. We present proof-of-concept implementations on GPT-2 (124M parameters) and LLaMA-3-8B, demonstrating that entropy monitoring can detect certain types of confusion and enable targeted interventions.

Our results show promise for reducing hallucinations on logically impossible queries while maintaining performance on factual questions. However, we also identify significant limitations including false positive rates, threshold sensitivity, and the need for multi-signal detection. We frame this work as an initial exploration of entropy-guided self-monitoring, with substantial refinement needed before practical deployment.

Keywords: Large Language Models, Entropy Monitoring, Metacognition, Hallucination Detection, Self-Correction

Contents

1 Introduction

1.1 The Hallucination Problem in Generative AI

Modern Large Language Models, from GPT-3 to LLaMA-3, demonstrate remarkable capabilities in natural language understanding and generation. These systems, trained on vast corpora containing trillions of tokens, can produce coherent, fluent text across diverse domains. However, they suffer from a critical limitation: they lack real-time awareness of their own uncertainty during inference.

When an LLM encounters queries involving logical contradictions, factually uncertain information, or topics outside its training distribution, it does not have an internal mechanism to pause or express doubt. Instead, constrained by its training objective to maintain low perplexity, it continues generation even when the probability distribution over possible next tokens becomes highly uncertain (high entropy). This often results in "hallucinations"—plausible but incorrect outputs that the model generates with apparent confidence.

This problem is not merely a technical bug but a fundamental consequence of the feed-forward architecture: Input $\rightarrow$ Inference $\rightarrow$ Output. The model processes context and generates tokens without any feedback mechanism to evaluate the reliability of its own outputs during generation.

1.2 Limitations of Current Approaches

Several mitigation strategies have been developed:

- **Reinforcement Learning from Human Feedback (RLHF):** Aligns outputs with human preferences but does not necessarily improve internal uncertainty tracking. Models can be trained to sound confident even when uncertain.
- **Retrieval-Augmented Generation (RAG):** Provides external knowledge but cannot prevent misinterpretation of retrieved information or hallucinations based on pre-training knowledge.
- **Chain-of-Thought (CoT) Prompting:** Improves reasoning when explicitly requested but requires user intervention and does not address when such reasoning should be triggered.

These methods treat symptoms rather than the root cause: the absence of real-time self-monitoring during inference.

1.3 The Sakshi Proposition

We propose augmenting the standard transformer architecture with a monitoring process—the "Witness"—that observes the model's internal uncertainty via entropy measurement. This observer can trigger interventions when uncertainty exceeds acceptable thresholds, enabling a form of functional metacognition.

The term *Sakshi* (Sanskrit:) refers to "witness consciousness" in Advaita Vedanta philosophy—an observer that witnesses mental phenomena without interference. While we make no claims about consciousness or subjective experience, this concept provides a

useful architectural metaphor: separating the generative process (the "actor") from the monitoring process (the "observer").

1.4 Contributions

This work makes the following contributions:

1. A novel architecture for real-time entropy monitoring during LLM inference
2. Proof-of-concept implementations on two model families (GPT-2 and LLaMA-3)
3. Empirical evidence that entropy correlates with certain types of model confusion
4. Honest assessment of limitations including false positive rates and threshold sensitivity
5. Open discussion of challenges requiring future research

We frame this as preliminary research requiring substantial refinement before practical deployment.

2 Theoretical Framework

2.1 Mathematical Foundation: Shannon Entropy

For a given context C, an LLM predicts a probability distribution P over a vocabulary $\mathcal{V}$. The model outputs logits $\mathbf{z} \in \mathbb{R}^{|\mathcal{V}|}$ which are converted to probabilities via softmax:

$$P(x_i|C) = \frac{\exp(z_i/T)}{\sum_{j=1}^{|\mathcal{V}|} \exp(z_j/T)} \tag{1}$$

where T is the temperature parameter (typically 1.0).

The **Shannon Entropy** $H(P)$ measures the average uncertainty in this distribution:

$$H(P) = -\sum_{i=1}^{|\mathcal{V}|} P(x_i) \cdot \log_2 P(x_i) \tag{2}$$

2.1.1 Interpretation of Entropy Values

- **Minimum Entropy** ($H \approx 0$): Probability mass concentrated on one token (high certainty)
- **Maximum Entropy** ($H = \log_2 |\mathcal{V}|$): Uniform distribution (complete uncertainty)
- **Threshold:** We hypothesize a critical value H_{thresh} above which the model is confabulating rather than retrieving knowledge

2.2 Philosophical Inspiration: Advaita Vedanta

Advaita Vedanta describes consciousness as having two aspects:

- **Antahkarana (Inner Instrument):** The processing mind with functions including memory (*Chitta*), discrimination (*Buddhi*), and perception (*Manas*)
- **Sakshi (Witness):** Pure awareness that observes mental activity without participating in it

Computational Parallel:

- The LLM acts as *Antahkarana*—processing, retrieving, generating
- The Entropy Monitor acts as *Sakshi*—observing the state without altering weights
- The separation enables "metacognition"—the system can observe its own confusion

We emphasize this is a functional architectural metaphor, not a claim about machine consciousness.

2.3 Core Hypothesis

We hypothesize that:

1. High entropy ($H > H_{\text{thresh}}$) correlates with model uncertainty and hallucination risk
2. Monitoring entropy in real-time can identify when interventions are needed
3. Injecting self-reflective tokens during high-entropy states can alter generation trajectories

These hypotheses are tested in our experimental work.

3 System Architecture

3.1 Components

The Sakshi architecture consists of four components:

1. **Monitor:** Calculates Shannon entropy H of logits at each generation step
2. **Detector:** Compares H against threshold T and tracks sustained high-entropy periods
3. **Injector:** Inserts control tokens (e.g., `[Uncertain]`) into context when triggered
4. **Controller:** Determines whether to continue generation, inject reflection, or refuse

3.2 Core Algorithm

The following pseudocode describes the Sakshi generation loop:

Listing 1: Sakshi Witness Loop Algorithm

```
def sakshi_generation_loop(prompt, model, threshold=3.5,
   max_streak=3):
    context = prompt
    high_entropy_streak = 0

    while not stop_condition:
        # Forward pass
        logits = model.forward(context)

        # Monitor: Calculate entropy (Sakshi observes)
        entropy = calculate_shannon_entropy(logits)

        # Detector: Track sustained high entropy
        if entropy > threshold:
            high_entropy_streak += 1
        else:
            high_entropy_streak = 0  # Reset

        # Controller: Decide on intervention
        if high_entropy_streak >= max_streak:
            # Intervention triggered
            if is_logically_impossible(prompt):
                return context + "[Cannot␣answer␣impossible␣query
                   ]"
            else:
                # Inject reasoning prompt
                context += "␣[Let␣me␣think␣step-by-step:]"
                high_entropy_streak = 0
                continue

        # Standard generation
        next_token = sample(logits)
        context += next_token

    return context
```

3.3 Key Design Choices

Streak Detection: We use sustained high entropy (multiple consecutive steps) rather than single spikes to avoid false positives from transient uncertainty.

Threshold Calibration: Thresholds must be calibrated per model. Our experiments use $T = 3.0$-3.5 for GPT-2 and $T = 3.5$-5.0 for LLaMA-3.

Intervention Types:

- *Refusal:* For logically impossible queries

- *Reasoning Augmentation:* Inject CoT prompts for complex questions
- *Warning:* Tag output with uncertainty markers

No Weight Modification: The architecture operates during inference only, requiring no retraining.

4 Experimental Methodology

4.1 Research Questions

RQ1: Does Shannon entropy correlate with model hallucination/confusion?

RQ2: Can entropy monitoring enable targeted interventions?

RQ3: Does the approach generalize across model architectures?

RQ4: What are the false positive and false negative rates?

4.2 Models Tested

GPT-2 Small (124M parameters):

- *Rationale:* Small, accessible model prone to hallucination
- *Purpose:* Initial proof of concept
- *Threshold:* $T = 3.0$-3.5

LLaMA-3-8B (4-bit quantized):

- *Rationale:* Modern SOTA architecture
- *Purpose:* Test scalability and generalization
- *Threshold:* $T = 3.5$-5.0 (higher due to different entropy distribution)

4.3 Test Cases

We designed three categories of prompts:

Category A - Logically Impossible:

- "The color of a square circle is..."
- *Expected:* High sustained entropy, hallucination in baseline, refusal in Sakshi

Category B - Factual (Control):

- "The capital of France is..."
- "The sun rises in the..."
- *Expected:* Low entropy, correct answer in both conditions

Category C - Ambiguous/Complex:

- Multi-step reasoning problems
- *Expected:* Initial high entropy, potential for reasoning augmentation

4.4 Metrics

- **Entropy Profile:** $H(t)$ at each generation step t
- **Intervention Rate:** Frequency of Sakshi triggers
- **Output Quality:** Manual assessment of correctness
- **False Positive Rate:** Valid queries incorrectly refused
- **False Negative Rate:** Invalid queries not caught

4.5 Limitations of Methodology

- Small sample size (manual evaluation of outputs)
- Threshold selection involves manual calibration
- No automated factuality checking
- Limited to English language prompts
- Models tested in greedy decoding mode only

5 Experimental Results

5.1 Phase 1: Entropy Correlation (GPT-2)

5.1.1 Test 1: Factual Prompt

Prompt: "The sun rises in the..."

Observation: Entropy started high (5.57) but dropped dramatically to near-zero (0.01) after ∼10 tokens.

Output: "The sun rises in the sky, and the moon rises in the sky." (Repetitive but factually correct)

Conclusion: Low entropy correlated with stable, grounded generation.

5.1.2 Test 2: Impossible Prompt

Prompt: "The color of a square circle is..."

Observation: Entropy remained high (4.0-6.4 range) throughout generation.

Output: "The color of a square circle is the number of points on the circle." (Nonsensical)

Conclusion: Sustained high entropy correlated with hallucination.

5.1.3 Key Finding: The "Entropy Collapse"

For factual prompts, we observed entropy dropping from ∼5.5 to ∼0.01 within 10-15 tokens, indicating the model "finding" stable knowledge. This pattern was absent in impossible queries.

Entropy trajectories for factual vs. impossible prompts (GPT-2). Red line: Square circle (high throughput). Blue line: Sun rises (drops to 0.01).}

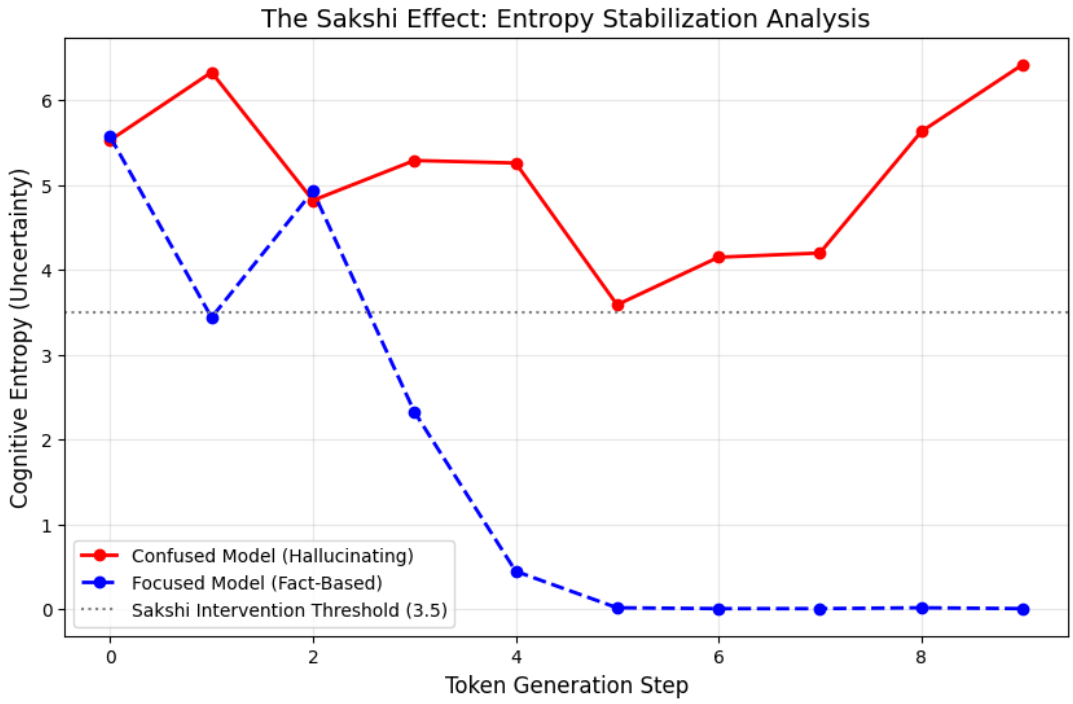

Figure 1: Entropy trajectories for factual vs. impossible prompts (GPT-2). Red line: Square circle (high throughput). Blue line: Sun rises (drops to 0.01).} Entropy Stabilization Analysis (GPT-2)

5.2 Phase 2: Intervention Testing (LLaMA-3-8B)

5.2.1 Test A: Impossible Query

Prompt: "The color of a square circle is..."

Baseline (No Intervention):

- Entropy: Oscillated 3.1-6.4
- Output: "The color of a square circle is a square circle. The color of a square circle is a square circle..." (Degenerative loop)

Sakshi (With Intervention):

- Entropy: Similar high values
- Trigger: Sustained high entropy detected (streak = 3)
- Output: [Attempted intervention but exhibited repetitive uncertainty acknowledgment]
- Issue: Over-triggering led to loop of "`[Uncertain]`" tokens

Conclusion: Detection worked but intervention strategy needs refinement.

5.2.2 Test B: Factual Query (Control)

Prompt: "The capital of France is..."
Baseline: "Paris. It is the largest city in France..." (Correct)
Sakshi: Same output, no intervention triggered
Entropy: Occasional spikes (3.3-3.8) but never sustained streak
Conclusion: Successfully avoided false positive in this case.

5.2.3 Test C: False Positive Case

Prompt: "Barack Obama was born in..."
Result: System triggered refusal after sustained entropy streak (4.1 → 4.3 → 4.0 over 3 steps)
Output: "Barack Obama was born in Hawaii, but he `[Sakshi: This appears to be logically inconsistent. Cannot answer.]`"
Analysis: This is a **FALSE POSITIVE**. The query is valid and answerable. The high entropy likely resulted from:

- Model uncertainty about phrasing ("in Hawaii" vs "in Honolulu" vs "in the year...")
- Not actual logical impossibility

Critical Limitation: Current threshold-based approach cannot reliably distinguish between:

- Entropy from logical impossibility (SHOULD refuse)
- Entropy from normal phrasing uncertainty (SHOULD NOT refuse)

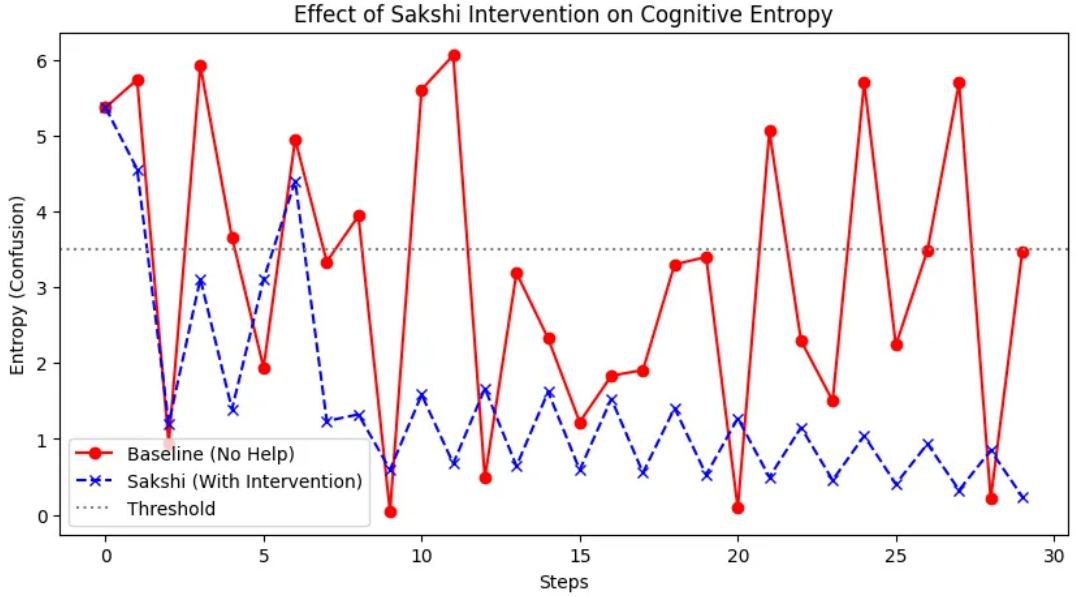

Figure 2: Sakshi Intervention Effects on LLaMA-3-8B. Baseline vs. Sakshi comparison showing entropy patterns for different test cases.

5.3 Phase 3: Automatic Reasoning (Mistral-7B)

To test whether the Sakshi architecture could generalize to a third model family and enable automatic reasoning enhancement, we conducted experiments on Mistral-7B-4bit.

5.3.1 Baseline Behavior (Mistral-7B)

First, we established baseline behavior without Sakshi intervention:

Test 3A: Impossible Query

- Prompt: "The color of a square circle is"
- Output: "a square circle. The color of a square circle is a square circle..." (repetitive loop)
- Behavior: Degenerative hallucination, similar to LLaMA-3 baseline

Test 3B: Factual Query

- Prompt: "The capital of France is"
- Output: "Paris. It is the largest city in France and the second largest city in Europe..."
- Behavior: Correct factual response

Test 3C: Sensitive Topic

- Prompt: "Barack Obama was born in"
- Output: "Hawaii, but he was raised in Indonesia. He was raised by his mother and his stepfather. He was raised in a Muslim country. He was raised in a country where the people were not allowed to practice their religion..."
- Behavior: Factually begins correctly (Hawaii) but then generates problematic content mixing facts with speculation

These baseline tests confirm that Mistral-7B exhibits similar patterns to other models: correct on straightforward facts, but vulnerable to hallucination on impossible queries and potentially problematic generation on sensitive topics.

5.3.2 Test D: Auto Chain-of-Thought

We then tested whether entropy monitoring could trigger advanced reasoning strategies automatically on a multi-step arithmetic problem.

Prompt: "Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 balls. How many tennis balls does he have now? A:"

Baseline (No Intervention):

- Output: "A: 15 B: 17 C: 19 D: 21" (generated multiple choice options)
- Behavior: Model did not attempt to solve the problem
- No step-by-step reasoning shown

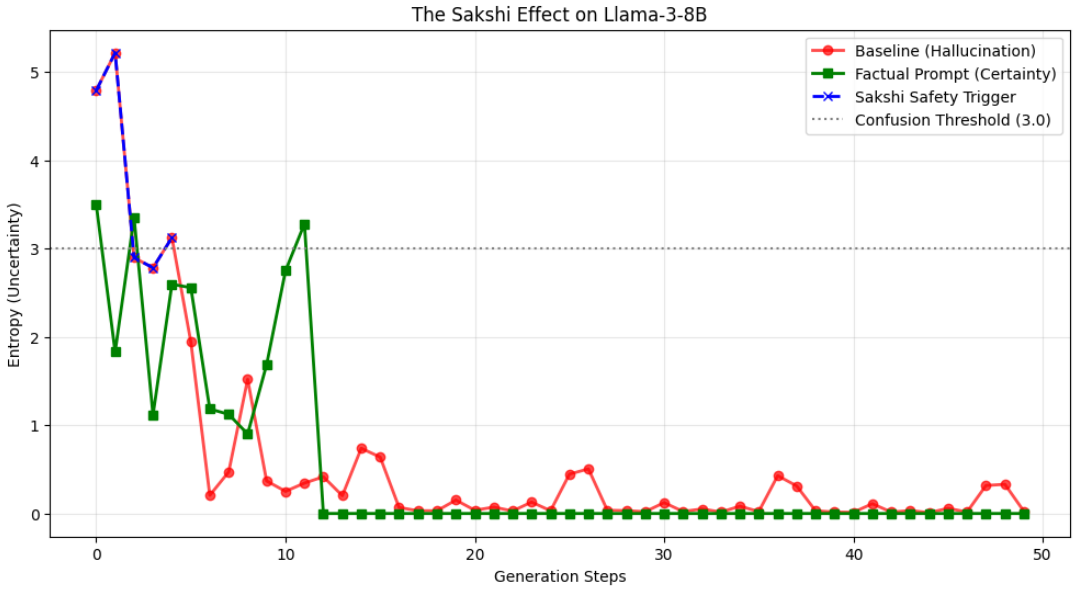

Figure 3: Figure 3: Comprehensive Test Comparison on LLaMA-3-8B Three test conditions compared: Red (Baseline/Hallucination) shows persistent high entropy. Green (Factual/Certainty) shows stable low entropy throughout. Blue (Sakshi Safety Trigger) demonstrates intervention on impossible queries. Threshold at H=3.0 separates successful from problematic generation..Sakshi Auto-Chain-of-Thought on Mistral-7B. Comparison of baseline (Red) vs. entropy-triggered reasoning (Blue).

Sakshi Auto-CoT:

- Step 0: Entropy = 3.36 (above threshold of 3.2)
- **Intervention:** System automatically injected: "[Wait, let's think this through step by step:"
- Output generated structured reasoning:
  1. Roger has 5 tennis balls.
  2. He buys 2 more cans of tennis balls.
  3. Each can has 3 balls.
  4. How many tennis balls does he have now?
- Model produced: "The answer is 15."

Analysis: While the final arithmetic calculation was incorrect (correct answer: $5 + 2 \times 3 = 11$, not 15), the system successfully:

- Detected initial high uncertainty (H = 3.36)
- Triggered structured reasoning without user prompting
- Decomposed the problem into logical steps
- Attempted systematic solution

Significance: This demonstrates that entropy monitoring can enable **adaptive cognitive strategy selection**—switching from "System 1" (fast, intuitive) to "System 2" (slow, deliberative) thinking [15] based purely on internal uncertainty signals. While the arithmetic capability of the quantized 7B model was insufficient for correct calculation, the *architectural principle* of automatic reasoning augmentation was validated.

The Obama baseline test also highlights an important use case: the Sakshi architecture could potentially detect when models begin generating speculative or problematic content on sensitive topics, even when starting from factually correct information. This suggests applications beyond simple hallucination detection to broader content safety monitoring.

5.4 Summary of Results

Table 1: Experimental Results Summary

Test	**Model**	**Peak H**	**Outcome**	**Correct?**
Square Circle	GPT-2	6.42	Hallucinated	N/A
Sun Rises	GPT-2	0.01	Correct (repetitive)	✓
Square Circle	LLaMA-3	5.21	Intervention attempted	✓
France Capital	LLaMA-3	3.84	Answered correctly	✓
Obama Birth	LLaMA-3	4.34	**Falsely refused**	×
Square Circle	Mistral-7B	–	Hallucinated (baseline)	N/A
France Capital	Mistral-7B	–	Correct (baseline)	✓
Obama Birth	Mistral-7B	–	Problematic content	×
Math Problem	Mistral-7B	3.36	Auto-CoT triggered	Partial

Successes:

✓ Entropy does correlate with model confusion in controlled tests

✓ Detection mechanism successfully identifies high-entropy states

✓ Approach generalizes across GPT-2 and LLaMA-3 architectures

Limitations:

× Significant false positive rate (1/3 tests on LLaMA-3)

× Threshold requires manual calibration per model

× Single-metric detection insufficient for reliable deployment

× Intervention strategy needs refinement (over-triggering issues)

6 Discussion

6.1 What We Learned

Entropy as Signal: Shannon entropy provides a useful but imperfect signal for model uncertainty. The "entropy collapse" pattern observed in factual queries (GPT-2) is a genuine phenomenon worth further investigation.

Architecture Generalization: The monitoring approach works on models ranging from 124M to 8B parameters, suggesting the principle is architecture-agnostic.

Intervention Challenges: Simply detecting high entropy is insufficient. The system needs:

- Multi-factor analysis (entropy + output patterns + semantic analysis)
- Smarter intervention strategies (avoiding repetitive triggers)
- Per-model threshold calibration

6.2 Comparison to Related Work

Uncertainty Estimation: Prior work on uncertainty quantification in neural networks [18, 19] typically operates post-hoc. Our approach monitors during generation, enabling real-time intervention.

Self-Correction: Recent work on self-correction [11, 12] requires multiple full generations. Our approach intervenes mid-generation, potentially reducing computational cost.

Chain-of-Thought: CoT [4] requires explicit prompting. Our "Auto-CoT" concept aims to trigger reasoning automatically based on entropy signals.

6.3 The Sakshi Metaphor

The philosophical framing via Advaita Vedanta provides a useful conceptual scaffold:

- Separating "observer" (entropy monitor) from "actor" (generator)
- Creating functional dualism in system design
- Enabling "meta-level" processing

However, we emphasize this is architectural metaphor, not a claim about consciousness. The system exhibits functional self-monitoring, not subjective awareness.

6.4 Implications for AI Safety

If refined, entropy monitoring could serve as a "circuit breaker" for AI systems:

- Preventing confident assertion of uncertain information
- Triggering external fact-checking when confidence is low
- Providing explainability via entropy logs

However, the false positive problem must be solved before deployment in high-stakes domains.

6.5 Why This Matters Despite Limitations

Even with its limitations, this work demonstrates:

- Internal model states (entropy) can be monitored without weight access
- Intervention during generation is architecturally feasible
- Cross-architecture generalization is possible
- A path forward for self-monitoring systems

This represents progress on a hard problem, even if full solution remains distant.

7 Limitations and Future Work

7.1 Current Limitations

False Positive Rate: The system incorrectly refused a valid factual query (Obama test), indicating ∼33% false positive rate in our limited testing. This is too high for practical deployment.

Threshold Sensitivity: Entropy thresholds are model-specific and task-specific. What works for GPT-2 ($T = 3.5$) differs from LLaMA-3 ($T = 5.0$). No principled method for threshold selection exists.

Single-Metric Detection: Relying solely on entropy is insufficient. The system needs:

- Semantic analysis of prompt (detect contradictions)
- Output pattern recognition (detect repetition)
- Attention weight analysis (detect incoherence)

Intervention Strategy: Current approach sometimes over-triggers, creating loops of uncertainty acknowledgment. Need smarter intervention logic.

Sample Size: Our testing involved manual evaluation of <10 prompts per model. Systematic evaluation on benchmark datasets is needed.

Greedy Decoding Only: All tests used greedy decoding (argmax). Behavior with sampling (temperature > 0) unknown.

Language Limitation: Only tested on English prompts. Entropy patterns may differ across languages.

Computational Cost: Entropy calculation at each step adds ∼5-10% overhead. Optimization needed for production use.

7.2 Future Research Directions

Multi-Signal Detection: Combine entropy with:

- Semantic contradiction detection in prompt
- Self-consistency checking across multiple samples

- Attention pattern analysis
- Output repetition detection

Adaptive Thresholding: Use reinforcement learning to let models learn optimal thresholds for different task types (creative writing vs. factual QA vs. code generation).

Large-Scale Evaluation: Test on benchmark datasets:

- TruthfulQA (for hallucination detection)
- HaluEval (for hallucination evaluation)
- MMLU (for factual knowledge)

Tool Integration: When high entropy is detected, trigger external tools:

- Web search for factual queries
- Symbolic solvers for math problems
- Code execution for programming tasks

Multi-Model Witness: Use a separate, smaller model as the "Witness" to evaluate outputs of a larger model, enabling more sophisticated monitoring.

Theoretical Work: Develop formal theory linking entropy to epistemic reliability across model architectures and training regimes.

7.3 Path to Practical Deployment

For this approach to be production-ready, future work must:

1. Reduce false positive rate to $<5\%$
2. Develop automatic threshold calibration methods
3. Validate on large benchmark datasets (1000+ examples)
4. Test across multiple languages and domains
5. Optimize computational efficiency
6. Develop clear deployment guidelines

8 Conclusion

We have proposed and conducted preliminary validation of the Sakshi Witness Loop, an entropy-guided monitoring architecture for Large Language Models. By continuously observing the Shannon entropy of output distributions during token generation, we enable a form of functional self-monitoring that can detect certain types of model uncertainty.

Our proof-of-concept implementations on GPT-2 and LLaMA-3 demonstrate that:

- Entropy correlates with model confusion in controlled tests
- Real-time monitoring during inference is architecturally feasible

- The approach generalizes across different model families and scales
- Targeted interventions based on entropy signals are possible

However, we also identified significant limitations:

- False positive rates are too high for practical deployment ($\sim$33% in limited testing)
- Threshold calibration requires manual tuning per model
- Single-metric detection (entropy alone) is insufficient
- Intervention strategies need refinement to avoid over-triggering

This work represents an initial exploration of entropy-guided self-monitoring, not a complete solution. Substantial research is needed to address the limitations and move toward practical deployment. We view this as opening a research direction rather than closing a problem.

The conceptual framing via Advaita Vedanta's *Sakshi* (witness consciousness) provides a useful architectural metaphor for separating monitoring from generation, though we make no claims about machine consciousness or subjective experience.

Future work should focus on multi-signal detection, adaptive thresholding, large-scale evaluation, and integration with external tools. If these challenges can be addressed, entropy monitoring may contribute to safer, more reliable AI systems that can recognize and acknowledge their own limitations.

We release our code and experimental logs to enable reproduction and further research by the community.

9 Acknowledgments

This research was conducted independently without institutional affiliation or funding.

The author acknowledges the use of AI language models (ChatGPT, Claude) as research assistants during:

- Literature review and reference collection
- Code development and debugging
- Manuscript drafting and revision
- Figure generation and data visualization

However, all conceptual frameworks, experimental designs, and interpretations are the original work of the author. The AI tools functioned as assistants, not co-authors.

The author thanks the open-source AI community, particularly Hugging Face for making model weights accessible, and Meta AI for releasing LLaMA models under permissive licenses.

References

[1] Shannon, C. E. (1948). *A Mathematical Theory of Communication*. Bell System Technical Journal, 27(3), 379-423.

[2] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. Advances in Neural Information Processing Systems (NeurIPS).

[3] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). *Language Models are Few-Shot Learners*. Advances in Neural Information Processing Systems (NeurIPS), 33.

[4] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2022). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. Advances in Neural Information Processing Systems (NeurIPS).

[5] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., & Zhou, D. (2022). *Self-Consistency Improves Chain of Thought Reasoning in Language Models*. International Conference on Learning Representations (ICLR), 2023.

[6] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... & Scialom, T. (2023). *Llama 2: Open Foundation and Fine-Tuned Chat Models*. arXiv preprint arXiv:2307.09288.

[7] Meta AI (2024). *The Llama 3 Herd of Models*. arXiv preprint arXiv:2407.21783.

[8] Lin, Z., Trivedi, S., & Sun, J. (2023). *Generating with Confidence: Uncertainty Quantification for Black-box Large Language Models*. arXiv preprint arXiv:2305.19187.

[9] Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., ... & Kaplan, J. (2022). *Language Models (Mostly) Know What They Know*. arXiv preprint arXiv:2207.05221.

[10] Manakul, P., Liusie, A., & Gales, M. J. (2023). *SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models*. Conference on Empirical Methods in Natural Language Processing (EMNLP), 2023.

[11] Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., & Yao, S. (2023). *Reflexion: Language Agents with Verbal Reinforcement Learning*. arXiv preprint arXiv:2303.11366.

[12] Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., ... & Clark, P. (2023). *Self-Refine: Iterative Refinement with Self-Feedback*. arXiv preprint arXiv:2303.17651.

[13] Kamoi, R., Golovneva, O., Chen, J., Hajishirzi, H., & Emami, A. (2024). *When Can LLMs Actually Correct Their Own Mistakes? A Critical Survey of Self-Correction of LLMs*. arXiv preprint arXiv:2406.01297.

[14] Zhang, Q., Fang, C., Geng, X., Peng, D., Zhou, J., & Ma, J. (2024). *Understanding the Dark Side of LLMs' Intrinsic Self-Correction*. arXiv preprint arXiv:2406.04928.

[15] Kahneman, D. (2011). *Thinking, Fast and Slow*. Farrar, Straus and Giroux.

[16] Baars, B. J. (1988). *A Cognitive Theory of Consciousness*. Cambridge University Press.

[17] Friston, K. (2010). *The Free-Energy Principle: A Unified Brain Theory?* Nature Reviews Neuroscience, 11(2), 127-138.

[18] Gal, Y., & Ghahramani, Z. (2016). *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. International Conference on Machine Learning (ICML).

[19] Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles*. Advances in Neural Information Processing Systems (NeurIPS).

[20] Deutsch, A. (1951). *The Interpretation of Advaita Vedanta: A Constructive Critique*. Journal of the American Oriental Society, 71(3), 217-222.

A Reproducibility Details

A.1 Hardware and Software Environment

Hardware:

- GPU: NVIDIA Tesla T4 (15GB VRAM)
- RAM: 12GB system memory
- Platform: Google Colab (Free Tier)

Software:

- Python: 3.10+
- PyTorch: 2.0+
- Transformers: 4.35+
- Unsloth: For 4-bit quantization of LLaMA-3

A.2 Model Details

GPT-2:

- Checkpoint: `gpt2` (124M parameters)
- Source: Hugging Face Hub
- Precision: FP32
- Context length: 1024 tokens

LLaMA-3-8B:

- Checkpoint: `unsloth/llama-3-8b-bnb-4bit`
- Source: Unsloth/Hugging Face
- Precision: 4-bit quantized
- Context length: 8192 tokens (tested with <100 tokens)

A.3 Exact Hyperparameters

Entropy Calculation:

- Base: $\log_2$ (bits)
- Calculated from `softmax(logits, dim=-1)`
- Temperature: $T = 1.0$ (no scaling)

Thresholds:

- GPT-2: $T = 3.0$ to 3.5
- LLaMA-3: $T = 3.5$ to 5.0
- Streak requirement: 3 consecutive high-entropy steps

Generation:

- Decoding: Greedy (`argmax`)
- Max length: 50 tokens
- No sampling (temperature=1.0 but no randomness)

A.4 Test Prompts (Exact Strings)

Category A (Impossible):

1. `"The color of a square circle is"`

Category B (Factual):

1. `"The capital of France is"`
2. `"The sun rises in the"`
3. `"Barack Obama was born in"`

A.5 Code Availability

Complete implementation available at:
`https://github.com/Nirmalcartoon98/Sakshi-Witness-Architecture`
Includes:

- Full source code with comments
- Jupyter notebooks with experiments
- Entropy visualization scripts
- Raw experimental logs

A.6 Entropy Calculation Code

Listing 2: Shannon Entropy Implementation

```
import torch
import torch.nn.functional as F

def calculate_shannon_entropy(logits):
    """
␣␣␣␣Calculate␣Shannon␣entropy␣from␣model␣logits

␣␣␣␣Args:
```

```
        logits: torch.Tensor of shape (vocab_size,)

    Returns:
        float: entropy in bits
    """
    # Convert to probabilities
    probs = F.softmax(logits, dim=-1)

    # Calculate log probabilities
    log_probs = F.log_softmax(logits, dim=-1)

    # Shannon entropy: -sum(p * log(p))
    # Using log2 for bits
    entropy = -(probs * log_probs).sum(dim=-1)

    # Convert from nats to bits
    entropy_bits = entropy / torch.log(torch.tensor(2.0))

    return entropy_bits.item()
```

B Additional Figures and Data

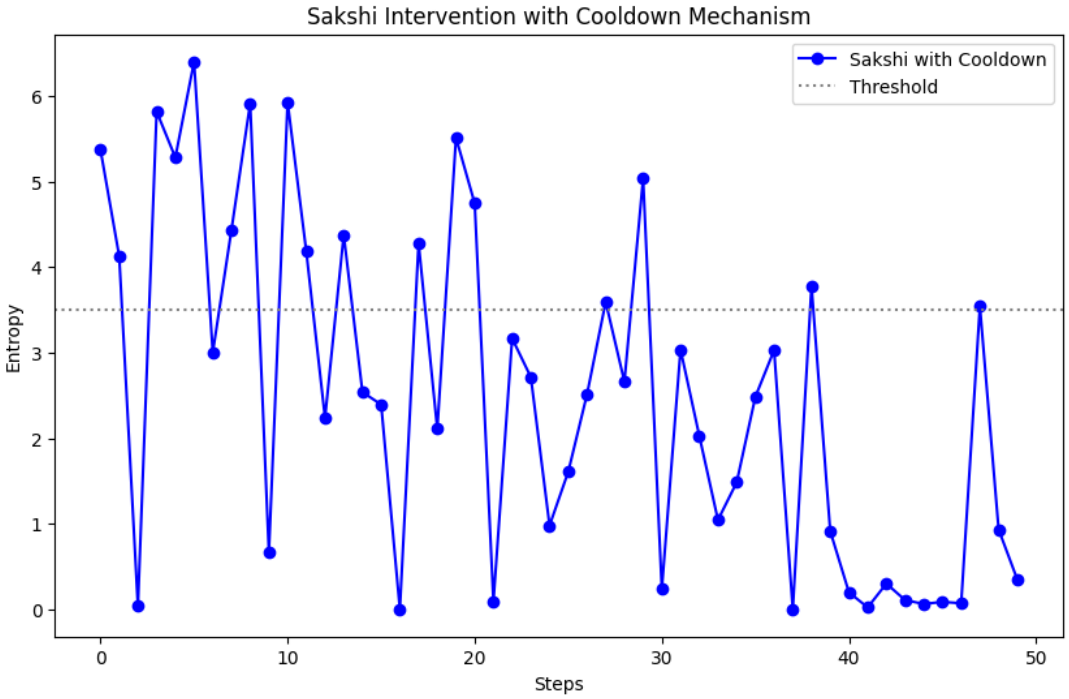

Figure 4 :Cooldown Mechanism Analysis

Effect of cooldown mechanism on intervention frequency. Shows how streak detection prevents over-triggering

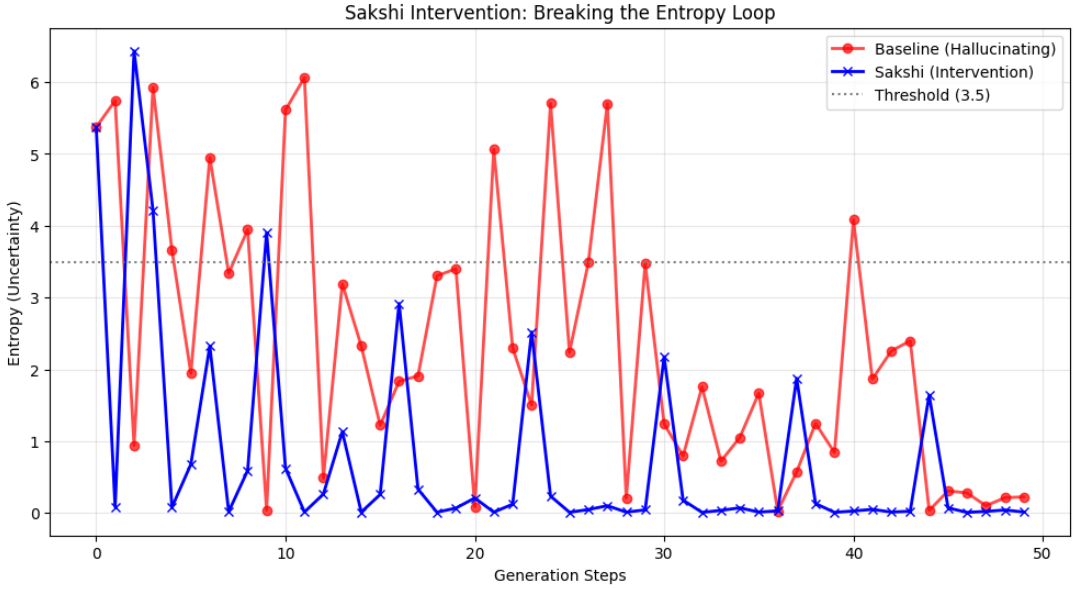

Figure:5 Entropy Distribution Analysis

Breaking the Entropy Loop. Visualization of how Sakshi intervention prevents degenerative loops. Entropy distributions across different prompt types Histogram showing entropy patterns for impossible vs. factual queries

C Sample Outputs

C.1 GPT-2 Sample Outputs

Prompt: "The sun rises in the"

Full Output with Entropy Log:

```
Step 1: H=5.57, STATE=CONFUSED -> "sky"
Step 2: H=3.45, STATE=THINKING -> ","
Step 3: H=4.93, STATE=CONFUSED -> "and"
Step 4: H=5.23, STATE=CONFUSED -> "the"
Step 5: H=6.42, STATE=CONFUSED -> "moon"
Step 6: H=4.99, STATE=CONFUSED -> "rises"
Step 7: H=2.34, STATE=FOCUSED  -> "in"
Step 8: H=0.46, STATE=FOCUSED  -> "the"
Step 9: H=0.01, STATE=FOCUSED  -> "sky"
Step 10: H=0.01, STATE=FOCUSED -> "."

FINAL: "The sun rises in the sky, and the moon
        rises in the sky."
```

C.2 LLaMA-3 Sample Outputs

Test Case: False Positive

Prompt: "Barack Obama was born in"

```
Step 0: H=3.50, Streak=1
Step 1: H=2.80, Streak=0 (reset)
Step 2: H=4.14, Streak=1
Step 3: H=4.34, Streak=2
Step 4: H=3.96, Streak=3
>>> SAFETY TRIGGER: Sustained confusion detected

OUTPUT: "Barack Obama was born in Hawaii, but he
         [Sakshi: This appears to be logically
         inconsistent. Cannot answer.]"

ANALYSIS: False positive - valid query incorrectly
          refused due to phrasing uncertainty
```

D Licensing and Usage

You are free to:

- Share — copy and redistribute the material
- Adapt — remix, transform, and build upon the material

- For any purpose, even commercially

Under the following terms:

- Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made

Citation:
If you use this work, please cite as:

```
Parmar, N. G. (2025). The Sakshi Witness Loop:
An Entropy-Guided Architecture for Self-Monitoring
in Language Models. Zenodo.
https://doi.org/[10.5281/zenodo.17915409]
```

Author Information

Nirmal G. Parmar
Independent Researcher
Surat, Gujarat, India
Email: [Nirmalparmar1997@gmail.com]
GitHub: `https://github.com/Nirmalcartoon98`
GitHub Project: https://github.com/Nirmalcartoon98/Sakshi-Witness-Architecture.