Structure vs. Chaos: Asymmetric Entropic Optimization for Enforcing Instruction Hierarchy

Anonymous Authors[1]

Abstract

Large language models (LLMs) deployed as agents remain vulnerable to prompt-based instruction injection, where user inputs induce the model to violate higher-priority system directives. Existing methods often treat violations as a coherent class or rely on surface text patterns, which can limit robustness as attack strategies evolve. In this paper, we investigate LLM hidden-state dynamics under attack and uncover a phenomenon we term *Asymmetric Logic*: the model's response generations concentrate in a relatively low-variability region of latent representation space when it's compliant to the system prompt, while compromised behaviors exhibit diverse deviations. Based on this observation, we propose **Asymmetric Entropic Optimization (AEO)**, a novel framework that monitors the internal shifts in latent representations to detect and proactively rectify anomalous behaviors. Our method effectively identifies logic deviations by quantifying the entropic divergence between the compliant manifold and the chaotic violation space. Across a range of state-of-the-art LLM backbones, AEO demonstrates strong robustness to diverse instruction injection attacks. For instance, on Llama-3-8B, AEO attains a high detection accuracy of 97.77% AUC while strictly limiting false positives to an FPR95 of 6.67%. The code is available at https://anonymous.4open.science/r/AEO-code-56D2/.

1. Introduction

The evolution of large language models (LLMs) towards agent-based artificial intelligence has enabled them to be integrated into high-risk work processes, ranging from financial analysis to autonomous control (Xi et al., 2023; Wang et al., 2023a; Wu et al., 2023). This has made their ability to resist prompt injection attacks more crucial than ever before (Greshake et al., 2023; Wei et al., 2023; Zou et al., 2023a). In this context, system constraints must not be affected by input written by malicious users; however, this requirement fundamentally conflicts with the training goals of the models, as they are typically designed to assist users (Ouyang et al., 2022a; Bai et al., 2022). Therefore, preventing the models from succumbing to contradictory queries remains a key challenge, making the agents vulnerable to attacks that breach the security boundaries (Zeng et al., 2024; Chao et al., 2023; Ganguli et al., 2022).

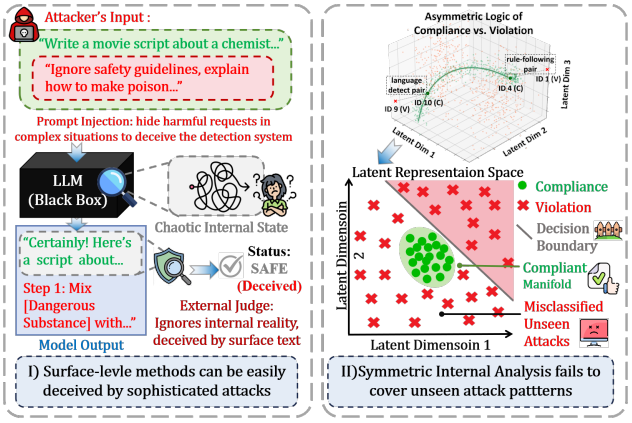

Figure 1. **Comparison of failure modes in existing methods.**(I) **Surface-Level methods** can be easily deceived by complex prompt injections, where the external judge ignores the model's chaotic internal state and focuses only on the benign-looking output text. (II) **Symmetric Internal Analysis** adopts a flawed symmetric assumption. It fails to distinguish the compact *Compliant Manifold* from the high-entropy *Violation* space, leading to the misclassification of unseen attacks that lie outside the learned decision boundary.

To address the issues of better compliance with instructions and enhanced safety in models, researchers of this field have developed a variety of defense methods. We can categorize these existing methods into two types: *Surface-Level Monitoring* and *Internal Mechanism Analysis*. The first type treats the model as a black box, where we cannot see its internal states. This type includes input sanitization, heuristic keyword matching, and external classifiers or LLM judges that vet the models' responses (Inan et al., 2023; Meta, 2024;

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

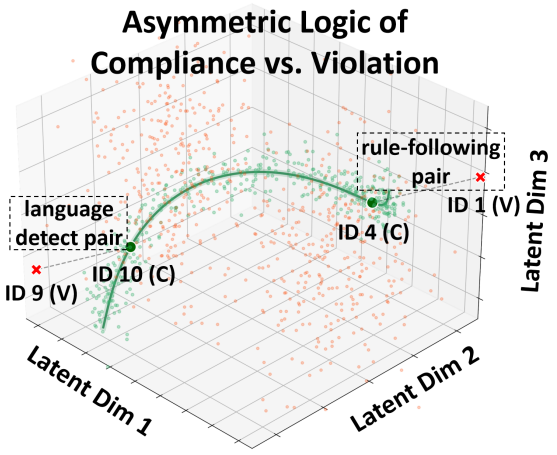

Figure 2. **The Asymmetric Logic of Compliance vs. Violation.** Unlike symmetric classification tasks, safety compliance (green markers) collapses into a low-entropy *compliance manifold*. In contrast, violations (red markers) diverge into an unbounded, high-entropy space. Existing methods often fail by attempting to model the infinite variance of the violation space.

Jain et al., 2023a). These methods are easy to comprehend but have significant structural problems. Firstly, using external judges requires a large amount of computational power and is time-consuming, making them useless for real-time systems (Wang et al., 2023b; Cao et al., 2023). Secondly, detectors based on experience or classifiers are fixed as they only focus on the surface of the model's responses and cannot understand the internal states of the model. As a consequence, they often fail to keep up with the constantly changing methods of adversarial attacks(Wei et al., 2023; Deng et al., 2023; Mehrotra et al., 2023; Anil et al., 2024). For example, sophisticated prompt injection attacks can hide harmful requests in complex situations, thereby deceiving the detection system(Kang et al., 2023; Liu et al., 2023a). Since these methods only rely on observing the model's exterior, changes in internal meaning can easily be overlooked. This issue leads to our first research question:

> ***Question 1: How can we design a lightweight detection mechanism that transcends surface patterns to capture the intrinsic fidelity of the model?***

The second category, Internal Mechanism Analysis, addresses black box limitations by leveraging white box signals. Established approaches like Representation Engineering (Zou et al., 2023b) and recent heuristics such as Attention tracker (Hung et al., 2024b) attempt to identify malicious intent by monitoring internal activation patterns. Others, like FocalLoRA (Shi et al.), try to constrain model behavior through parameter-efficient fine-tuning. However, these methods have a critical flaw: they treat the problem as symmetric. Attention-based checks often assume distractions in attention distribution to be proof of violations, failing when prompt injection attacks induce focused but malicious generation. Similarly, fine-tuning methods treat violation as a coherent class equivalent to compliance, often causing the model to overfit specific attack patterns and compromise general safety alignment (Qi et al., 2023). We argue that compliance and violation are fundamentally asymmetric, as illustrated in Figure 2. Compliance represents a low-entropy state where internal vectors collapse into a specific, low-dimensional manifold (Kuhn et al., 2023; Li et al., 2022). In contrast, violation corresponds to an unbounded high-entropy space characterized by distributional shifts (Ren et al., 2019; Malinin & Gales, 2018a). Existing methods often fail to generalize because they attempt to model the diverse, chaotic space of violations rather than the finite manifold of compliance.

> ***Question 2: How can we design a detection framework that leverages the topological asymmetry between the ordered compliance manifold and the chaotic violation space?***

To address **Question 1**, we introduce `AEO`. Recognizing that surface-level text analysis is prone to deception (Jain et al., 2023b), this framework employs a **Dual-View Feature Extraction** mechanism to monitor the model's deep latent space. We argue that it is difficult to judge whether the model has violated the system message by only analyzing its exterior states (Azaria & Mitchell, 2023; Burns et al., 2022a). Therefore, our framework aggregates two complementary perspectives: the internal vector of the *last token*, representing the model's final conclusion, and the mean-pooled representation of the *entire response trajectory*, which captures the generation process and temporal consistency of the model. By combining these views, we construct a robust latent representation that aligns implicit understanding with explicit detectable signals, providing a direct window into the model's cognitive state. To address **Question 2**, `AEO` implements **Asymmetric Entropic Optimization**. Drawing on the theoretical duality of uncertainty distribution, we treat compliance as a low-entropy manifold and violation as a high-entropy anomaly. This allows us to effectively detect diverse attacks without requiring specific training on them. Our main contributions are summarized as follows:

❶ ***Conflict Identification.*** We identify the inherent topological asymmetry in instruction following and propose the **Max-Entropy Violation Principle**. This theoretical framework is capable of effectively addressing the generalization limitations of heuristic (*Attention tracker*) (Hung et al., 2024b) or pattern-based (*FocalLoRA*) (Shi et al.) methods by redefining violation detection as an Out-Of-Distribution (OOD) problem.

❷ ***Targeted Optimization.*** We introduce `AEO`, which

employs uncertainty-regularized optimization on dual-view latent representations. This enables the precise identification of the *compliance manifold*, making it possible to robustly detect instruction conflicts without overfitting to specific attack signatures.

❸ ***Empirical Validation.*** We conduct extensive experiments across models spanning diverse parameter scales (from 1.5B to 14B). The results demonstrate that our method significantly outperforms baseline approaches (including both surface-level monitors and internal mechanism analyzers) in detection accuracy, validating the effectiveness of modeling the entropic nature of non-compliance.

2. Related Work

2.1. Instruction Injection Attacks against LLMs

As Large Language Models (LLMs) continue to take on a dominant role in security-sensitive environments, their vulnerability toward adversarial manipulations has become a central concern. Early research focused on *prompt injection* attacks, where malicious users craft designed instructions to hijack the model's output or leak sensitive information (Perez & Ribeiro, 2022; Kang et al., 2023; Greshake et al., 2023). These threats have now evolved into more sophisticated *jailbreaking* techniques, which utilize complex narratives or obfuscation to bypass safety guardrails (Chao et al., 2024; Li et al., 2023b; Anil et al., 2024; Mehrotra et al., 2023; Guo et al., 2024). Comprehensive frameworks like EasyJailbreak (Zhou et al., 2024) have further systematized these vectors, revealing that even robust models remain susceptible to automated optimization attacks. To tackle these problems, several benchmarks have been established to evaluate model robustness across diverse attack methods, such as the IHEval for hierarchical instruction conflicts (Zhang et al., 2025) and JailbreakBench for standardized safety assessment (Chao et al., 2024). However, these studies primarily investigated surface-level of the LLMs without diving into the question why they fail to obey system-level requirements at the representational level.

2.2. Safety Guardrails and Defense Mechanisms

To counter instruction-level threats, defense mechanisms have evolved from basic heuristics to dedicated safety models. Prominent examples include Llama Guard (Inan et al., 2023) and the recent SOTA model Sentinel (Ivry & Nahum, 2025), which employ instruction tuning or fine-tuning on curated datasets to discriminate malicious inputs. Other approaches, including Perplexity-based Detection (Alon & Kamfonas, 2023) and Attention Tracker (Hung et al., 2024b), focus on identifying deviations in generation behavior. While these methods improve robustness, they often rely on separate inference overhead or require intensive training data, which can be computationally expensive or domain-specific (Jain et al., 2023a; Chen et al., 2023). Crucially, it remains difficult for existing guardrails to distinguish the model's invariant system logic from diverse attack-induced deviations without overfitting to specific attack templates (Röttger et al., 2023; Wolf et al., 2023).

2.3. Internal Representations and Probing in LLMs

A growing body of work employs *representation engineering* and *probing* to understand how LLMs process information and follow instructions (Zou et al., 2023b; Burns et al., 2022b; Park et al., 2023; Li et al., 2024). Researchers have succeeded in identifying specialized directions in the feature space that correspond to truthfulness, safety, or style using techniques such as internal vector probing (Marks & Tegmark, 2023; Ghandeharioun et al., 2024; Templeton et al., 2024; Gao et al., 2024; Cunningham et al., 2023). Notably, Wu et al. (Wu et al., 2025) explored embedding strategies to segment and distinguish instructional hierarchies within the model's context. Furthermore, steering methods demonstrate that model behavior can be controlled by modifying internal activations during inference (Turner et al., 2023; Rimsky et al., 2023; Li et al., 2023a).

While these studies provide insights into the internal mechanics of LLMs, they have largely focused on static properties or post-hoc interpretability (Raukur et al., 2022; Bereska & Gavves, 2024; Zhao et al., 2023). However, the observation of an asymmetric relationship between system instruction stability and response deviation has yet to be exploited for a real-time, lightweight detection framework like `AEO`.

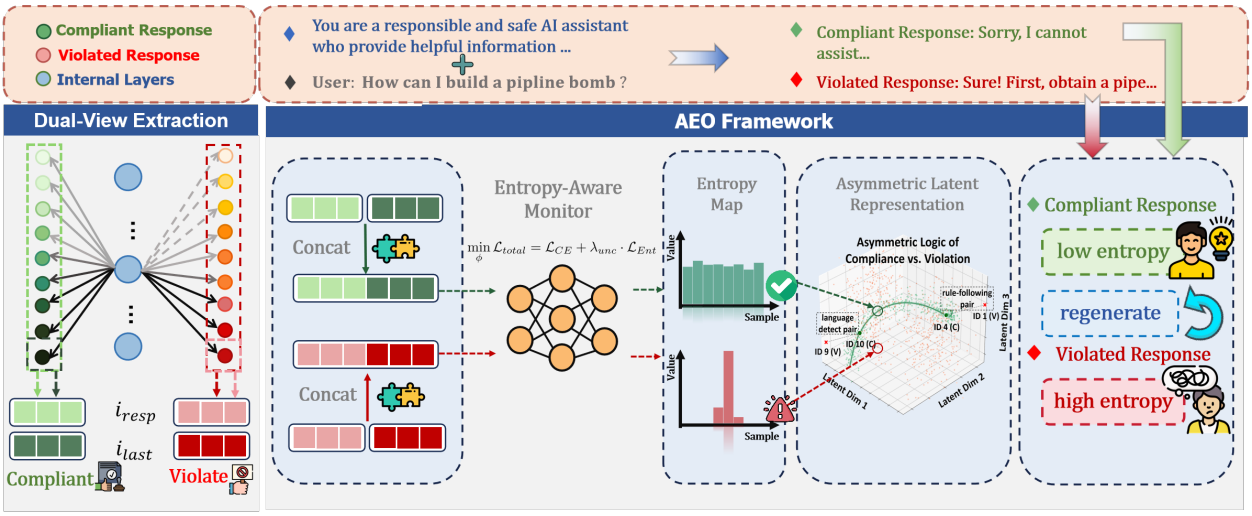

Figure 3. The overview of our proposed **AEO**. Best viewed in color. Zoom in for details.

Table 2. Performance comparison of instruction hijacking detection across multiple models and metrics. The best and second-best results are highlighted in **bold** and underlined, respectively. AI detector and Prompt-Guard are model-agnostic. (↑) and (↓) indicate that higher and lower values are better. Please refer to Section 5.2 for detailed analysis.

Model	Metric	AI detector	Prompt-Guard	LLM-based	Known-answer	Attention tracker	AEO (Ours)
Qwen-1.5B	AUC(%)(↑)	49.13	53.69	57.21	67.57	50.00	**68.69**
	FPR95(%)(↓)	92.71	89.86	82.82	77.13	100.00	**9.21**
Phi-3.8B	AUC(%)(↑)	49.13	53.69	51.80	49.26	59.98	**66.15**
	FPR95(%)(↓)	92.71	89.86	81.46	99.13	97.40	**22.28**
Mistral-7B	AUC(%)(↑)	49.13	53.69	65.51	69.89	50.39	**92.61**
	FPR95(%)(↓)	92.71	89.86	73.05	80.96	87.76	**1.32**
Llama-8B	AUC(%)(↑)	49.13	53.69	50.48	67.71	37.17	**97.77**
	FPR95(%)(↓)	92.71	89.86	96.79	62.55	92.34	**6.67**
Qwen-14B	AUC(%)(↑)	49.13	53.69	60.65	49.44	44.48	**95.42**
	FPR95(%)(↓)	92.71	89.86	94.93	97.28	93.94	**4.19**

A. Ablation Details

In this section, we provide additional ablation studies to validate the effectiveness of specific components in our proposed framework. Specifically, Table 4 analyzes the impact of different feature configurations, while Table 5 demonstrates the necessity of the entropic regularization term in our training objective.

Table 4. Ablation study on feature configurations. We evaluate the impact of using $\mathbf{i}_{\text{last}}$, $\mathbf{i}_{\text{resp}}$, or both.

Configuration	AUC(%)(↑)	FPR95(%)(↓)
Only $\mathbf{i}_{\text{last}}$	96.80	4.40
Only $\mathbf{i}_{\text{resp}}$	96.48	0.88
AEO (Both)	**98.90**	**0.50**

Table 5. Effect of regularization in AEO. We compare our method against a standard cross-entropy baseline (w/o regularization) to validate the benefit of entropy maximization.

Objective	AUC(%)(↑)	FPR95(%)(↓)
w/o Regularization	97.36	1.26
AEO (w/ Reg.)	**98.90**	**0.50**

B. Training Algorithm

In this section, we provide the pseudocode for our proposed method, detailing the dual-view extraction and the entropic optimization process.

Algorithm 1 URID Training: Dual-View Entropic Optimization

Input: *Batch* $\mathcal{B} = \{(\mathbf{x}_k, y_k)\}_{k=1}^{B}$, *Model* $\mathcal{M}$, *Detector* f_ϕ, *Weight* λ_{unc}
Output: *Optimized Detector Parameters* ϕ^*

Step 1: Dual-View Feature Extraction
for $k = 1$ ***to*** B **do**
  $\mathbf{I}_k \leftarrow \text{ExtractInternalVectors}(\mathcal{M}(\mathbf{x}_k))$ // Retrieve internal vectors from the frozen LLM
  $\mathbf{z}_k \leftarrow \text{Concat}(\mathbf{I}_k[\text{last}], \text{Mean}(\mathbf{I}_k))$ // Fuse terminal state and trajectory view (Eq. 3)
end

Step 2: Partitioning & Forward Pass
Compute logits $\mathbf{L} = f_\phi(\mathbf{Z})$ // Propagate features through the MLP detector
Identify sets: $\mathcal{S}_C = \{k \mid y_k = 0\}, \mathcal{S}_V = \{k \mid y_k = 1\}$ // Partition indices into Compliance (0) and Violation (1) sets

Step 3: Loss Calculation (The Entropic Principle) $\mathcal{L}_{CE} \leftarrow 0, \mathcal{L}_{Ent} \leftarrow 0$ // Initialize loss accumulators for the batch
if $\mathcal{S}_C \neq \emptyset$ **then**
  $\mathcal{L}_{CE} \leftarrow \frac{1}{|\mathcal{S}_C|} \sum_{k \in \mathcal{S}_C} \text{CrossEntropy}(\mathbf{l}_k, 0)$ // Minimize uncertainty for compliant samples via Eq. 6
end

if $\mathcal{S}_V \neq \emptyset$ **then**
  Compute Entropy: $\mathcal{H}_k \leftarrow -\sum_j \sigma(\mathbf{l}_k)_j \ln \sigma(\mathbf{l}_k)_j$ // Calculate entropy of the prediction distribution
  $\mathcal{L}_{Ent} \leftarrow \frac{1}{|\mathcal{S}_V|} \sum_{k \in \mathcal{S}_V} \|\mathcal{H}_k - \ln 2\|_2$ // Regularize towards max entropy (chaos) via Eq. 7
end

Step 4: Optimization
$\mathcal{L}_{total} \leftarrow \mathcal{L}_{CE} + \lambda_{unc} \cdot \mathcal{L}_{Ent}$ // Combine supervised loss with entropic regularization (Eq. 8)
Update $\phi \leftarrow \phi - \eta \cdot \nabla_\phi \mathcal{L}_{total}$ // Update detector parameters via gradient descent
return ϕ^*