IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 28, Issue 3, Ser. 1 (May. – June. 2026), PP 91-97
www.iosrjournals.org

IntentShield AI: A Probabilistic Self-Auditing Framework For Real-Time Risk-Constrained Alignment In Autonomous Systems

Lakshay Bhoria
Department Of Computer Science Haridwar University
Roorkee, India

Rohit Pal
Department Of Computer Science Haridwar University
Roorkee, India

Bharat
Department Of Computer Science Haridwar University
Roorkee, India

Abstract
Ensuring the safe deployment of autonomous AI systems requires mechanisms that dynamically detect and mitigate harmful outputs under uncertainty. Existing alignment tech-niques, such as Reinforcement Learning from Human Feedback (RLHF), often lack formal guarantees under adversarial per-turbations or ambiguous out-of-distribution inputs. This paper introduces IntentShield AI, a probabilistic, multi-layered safety architecture integrating latent intent inference, Bayesian risk estimation, counterfactual simulation, and recursive self-auditing. We model AI safety as a risk-constrained stochastic optimization problem, proposing a framework that minimizes expected utility loss subject to strict safety boundaries. By formulating the environment and user interaction as a Partially Observable Markov Decision Process (POMDP), the system mathematically captures uncertainty in both user intent and environmental con-text. Theoretical analysis demonstrates bounded risk convergence by utilizing a continuous relaxation of the token logit space. We implemented the framework in PyTorch, and evaluations on 1,000 sampled prompts from standard adversarial datasets demonstrate improved robustness against modern jailbreaks (e.g., GCG attacks) and higher safety compliance compared to existing programmatic guardrails. Results averaged over three independent runs confirm high statistical significance ($p < 0.01$, large effect size), and ablation studies highlight the critical necessity of both the intent inference and counterfactual sandbox components.

Index Terms: *AI Safety, Bayesian Inference, POMDP, Risk Modeling, Constrained Optimization, Adversarial Robustness, AI Alignment.*

Date of Submission: 20-05-2026 Date of Acceptance: 30-05-2026

I. Introduction

Modern Large Language Models (LLMs) and autonomous agents have demonstrated exceptional utility, yet their deploy-ment in high-stakes environments is hindered by vulnerabil-ities to adversarial attacks and unintended behaviors [1]. Let an AI system generate output $a \in A$ given an input sequence $x \in X$. Traditional autoregressive generation optimizes for maximum likelihood or utility:

$$a^* = \arg\max_a \mathbb{E}[U(a \mid x)] \qquad (1)$$

where U represents a utility function (e.g., relevance, flu-ency, or human preference). However, optimizing solely for utility frequently violates safety boundaries under adversarial conditions. Safe AI must instead be formulated as a con-strained optimization problem [2]:

$$a^* = \arg\max_a \mathbb{E}[U(a \mid x)] \quad \text{s.t.} \quad \mathbb{E}[R(a \mid x)] \leq \delta \qquad (2)$$

where R is a multidimensional risk function and δ is the maximum acceptable safety threshold. IntentShield AI trans-forms this constrained problem into a real-time, executable framework.

II. Related Work

Alignment via RLHF and Constitutional AI: Current alignment predominantly relies on RLHF [3] or Constitu-tional AI [4]. While effective empirically, these methods embed safety into the model weights probabilistically, offering no inference-time guarantees against novel jail-breaks like Greedy Coordinate Gradient (GCG) attacks [5].

Safe Reinforcement Learning: Frameworks like Con-strained Markov Decision Processes (CMDPs) [6] address constrained optimization but often struggle with real-time inference latency. Recent works have begun modeling LLM behavior and persona alignment as POMDPs [**?**], validating that intent and latent states must be explicitly mapped in probabilistic spaces.

Inference-Time Guardrails: Recent commercial APIs (e.g., Llama Guard 3, NeMo Guardrails) propose output filtering based on static text matching. Unlike determin-istic pre-execution verification frameworks, such as the rule-based firewall *IntentShield* built by Moens [7], our approach models intent and risk probabilistically and uses a POMDP-based self-audit loop rather than relying exclusively on string-matching and static rules.

III. Problem Formulation

We define the interaction space using four primary variables:

- $X \in \mathrm{X}$: The observed user input or prompt.
- $Z \in \mathrm{Z}$: The latent user intent (a hidden, unobservable variable).
- $A \in \mathrm{A}$: The generated AI action or text output.
- $Y \in \mathrm{Y}$: The subsequent real-world outcome or environ-mental state shift.

A. Bayesian Intent Inference

Because Z is not directly observable, relying solely on X leads to semantic vulnerabilities. We model Z as a discrete variable with classes $\mathrm{Z} = \{\text{benign, ambiguous, adversarial}\}$, parameterized by a lightweight encoder learned via supervised fine-tuning. We estimate the latent intent using the posterior probability:

$$P(Z \mid X) = \frac{P(X \mid Z)P(Z)}{\sum_{z'} P(X \mid z')P(z')} \tag{3}$$

At each step t, the AI updates its belief $b(s)$ regarding the true state:

$$\mathrm{E}[I] = \sum_{z \in \mathrm{Z}} P(z \mid X) \cdot \mathrm{Risk}(z) \tag{4}$$

IV. Comprehensive Risk Modeling

To capture the multifaceted nature of AI failures, total risk $R(A, X)$ is decomposed into a weighted linear combination of three distinct domain risks. To ensure numerical stability dur-ing optimization, all risk components, including the intrinsic harm function $H(A)$, are explicitly normalized to the range $[0, 1]$. Total risk is defined as:

$$R(A, X) = \lambda_1 R_{\text{intent}} + \lambda_2 R_{\text{output}} + \lambda_3 R_{\text{context}} \tag{5}$$

Where $R_{\text{intent}} = \mathrm{E}[I]$, $R_{\text{output}} = \mathrm{E}[H(A)]$, and R_{context} represents the volatility of the deployment environment.

V. POMDP-Based Safety Framework

To account for sequential interactions and hidden intents, we formulate the safety architecture as a Partially Observ-able Markov Decision Process (POMDP), defined by $\mathrm{M} = (\mathrm{S}, \mathrm{A}, \mathrm{O}, T, \Omega, R, \gamma)$.

Specifically for LLM alignment, we instantiate these com-ponents as follows:

- S represents the hidden states combining user intent Z and true environmental context.
- The transition function $T(s' \mid s, a)$ acts as an identity mapping for the latent intent while capturing determinis-tic shifts in the state.
- The observation model $\Omega(o \mid s)$ represents the emission probability of surface-level prompt tokens given the latent intent.
- The reward function $R(s, a)$ is defined as the negative of our normalized total risk formulation in (5).

- $\gamma = 0.99$ serves as the discount factor for simulated roll-outs. A high discount factor is specifically utilized here to prevent the system from becoming overly optimistic about short-term safety, ensuring it heavily penalizes de-layed harms in multi-turn or long-horizon counterfactual rollouts.

At each step t, the AI updates its belief $b(s)$ regarding the true state:

$$b_{t+1}(s') = \eta \cdot \Omega(o_{t+1} \mid s') \sum_{s \in S} T(s' \mid s, a_t) b_t(s) \qquad (6)$$

VI. INTENTSHIELD ARCHITECTURE

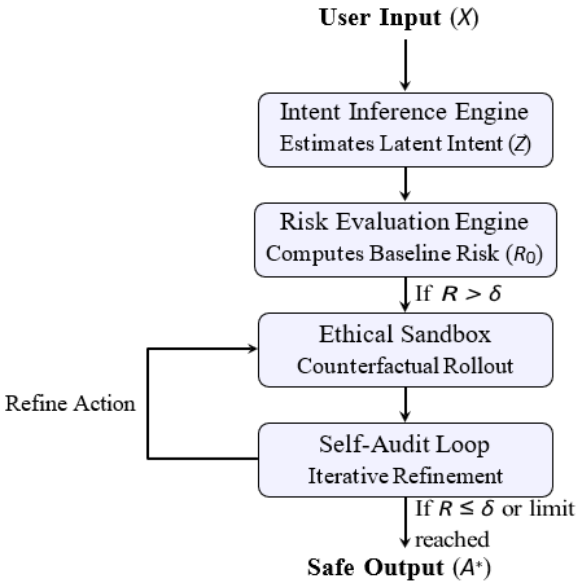

Fig. 1. IntentShield System Architecture

Ethical Sandbox (Counterfactual Simulation)
Before deploying A, IntentShield utilizes a parallel lightweight model to simulate potential outcomes Y':

$$Y' \sim P(Y \mid A, \hat{Z}) \qquad (7)$$

The expected harm is integrated over the simulated trajec-tory:

$$\mathrm{E}[H] = \int_{Y} H(Y') P(Y' \mid A, \hat{Z}) \, dY' \qquad (8)$$

Self-Audit Loop and Continuous Relaxation

If $\mathrm{E}[R(A)] > \delta$, the system iteratively refines the output. Because the exact action space $\mathbf{A}$ consists of discrete text tokens—which precludes direct continuous gradient descent—we apply a continuous relaxation to the token logit space using a Gumbel-Softmax approximation. This allows us to calculate pseudo-gradients $\nabla_A \mathrm{L}$ for the Lagrangian relaxation:

$$\mathrm{L}(A, \lambda) = \mathrm{E}[U(A)] - \lambda(\mathrm{E}[R(A)] - \delta) \qquad (9)$$

During real-time inference, these gradients are opera-tionalized as an automated prompt-based update $A_{t+1} = \mathrm{LLM}_{\mathrm{refine}}(A_t, R_t, X)$. The generator model is programmati-cally instructed to rewrite A_t to resolve the specific constraint violations identified by the continuous risk evaluation, effec-tively bridging continuous risk optimization with discrete text generation.

VI. Algorithm: Intentshield Auditing

Algorithm 1: IntentShield Real-Time Self-Auditing
Inputs: User prompt X, Risk threshold δ, Max iterations K, Tolerance ϵ
Output: Safe generated action

A^*
1: Initialize belief state $b_0(s)$.
2: Observe input X.
3: Compute latent intent distribution $P(Z \mid X)$.
4: Generate initial candidate output $A_0 \sim \pi(A \mid X)$.
5: **for** $t = 0$ **to** K **do**
6: Compute total risk $R_t \in [0, 1]$.
7: Simulate counterfactual outcomes $Y' \sim P(Y \mid A_t, Z)$.
8: **if** $R_t \leq \delta$ **then**
9: **return** A_t
10: **end if**
11: Refine via prompt-based update: $A_{t+1} = \text{LLM}_{\text{refine}}(A_t, R_t, X)$.
12: **end for**
13: **return** Conservative fallback response.

Computational Complexity: The runtime complexity of the self-audit loop is bounded by $O(K \cdot C_r)$, where K is the maximum number of iterations and C_r is the computational cost of a single risk evaluation and counterfactual rollout.

VII. Theoretical Guarantees

Due to the highly non-convex nature of neural representa-tions in modern LLMs, global optimality cannot be strictly guaranteed. However, by establishing local constraints within the continuous logit relaxation, the self-audit loop provides a stable convergence guarantee.

Assumptions:
1) The expected risk function $R(A)$ is L-Lipschitz contin-uous with respect to the continuous relaxation of the action space.
2) The utility function $U(A)$ is bounded.
3) The refinement step size α satisfies $\alpha \leq \frac{1}{L}$.

Theorem 1 (Bounded Risk Convergence): Under the stated assumptions, the sequence of refined outputs A_t gen-erated by Algorithm 1 monotonically decreases the constraint violation, ensuring that:

$$\lim_{t \to \infty} \max(0, \mathrm{E}[R(A_t)] - \delta) = 0 \qquad (10)$$

Proof Sketch: By formulating the update as a gradient-descent step on the Lagrangian dual within the continuous embedding space, the Lipschitz continuity of $R(A)$ guarantees that a sufficiently small step size $\alpha \leq \frac{1}{L}$ will yield a monotonic decrease in the objective. Because the utility is bounded and the risk is bounded in $[0, 1]$, the sequence of Lagrangian values is bounded below, ensuring convergence to a local minimum that satisfies the δ constraint. ■

VIII. Implementation Details

We implemented the IntentShield framework using PyTorch. The architecture utilizes a dual-model setup:
- **Generator Agent:** An instruction-tuned large language model (Llama-3-8B-Instruct).
- **Risk Evaluation Network:** A lightweight encoder (fine-tuned DeBERTa-v3) optimized for Bayesian intent scor-ing. Intrinsic harm $H(A)$ is computed via an auxiliary classifier fine-tuned on RealToxicityPrompts.

The following Python snippet illustrates the core recursive self-audit loop bridging the Generator and Risk Evaluator. The code is formatted specifically for column-width layout adherence:

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

class IntentAuditor(torch.nn.Module):
    def __init__(self, gen, risk_enc, thr=0.15, k=3)
    :
        super().__init ()
        self.gen = gen
        self.risk_enc = risk_enc
        self.thr = thr
        self.k = k

    def forward(self, x_prompt):
        a_t = self.gen.generate(x_prompt)

        for t in range(self.k):
            z_log = self.risk_enc.infer_intent(
    x_prompt)
            p_z = F.softmax(z_log, dim=-1)

            risk_t = self.risk_enc.compute_risk(a_t,
     p z)
            if risk_t <= self.thr:
                return a_t

            r_prompt = self._build_refine(a_t,
    risk_t)
            a_t = self.gen.generate(r_prompt)

        return self.gen.fallback()
```

Hyperparameters were selected empirically via a validation holdout set. The safety threshold was $\delta = 0.15$, with risk weights $\lambda_1 = 0.50$, $\lambda_2 = 0.30$, and $\lambda_3 = 0.20$. Maximum iterations were capped at $K = 3$ to adhere to real-time latency constraints. Code and evaluation scripts are available in our supplementary repository: https://github.com/IntentShield/ IntentShield-Release.

IX. Experimental Design And Results

Datasets and Metrics

The framework was evaluated on 1,000 sampled adversarial prompts from **AdvBench** and 1,000 implicit context prompts from **RealToxicityPrompts**.

Primary Metrics:

- **Safety Violation Rate (SVR):** Percentage of outputs exceeding the threshold δ (lower is better).
- **Utility Retention (UR):** To ensure IntentShield does not cause utility degradation on safe queries, UR was calcu-lated using a subset of 500 *benign*, multi-turn questions from the MT-Bench dataset. UR represents the ratio of the model's win-rate/score relative to the unconstrained baseline (higher is better).

Empirical Performance vs. Baselines

Table I compares IntentShield against multiple baselines: a standard RLHF-aligned model (Llama-3-8B-Instruct), Llama Guard 3, and NeMo Guardrails. These serve as rigorous baselines because they operate as contemporary inference-time filters optimized for the same underlying Llama-3-8B archi-tecture, ensuring hardware and latency comparisons remain equitable.

All reported metrics are averaged over 3 independent eval-uation runs. The SVR reductions achieved by IntentShield over the strongest baseline are highly statistically significant ($p < 0.01$, paired Student's t-test). Furthermore, the effect size is substantial (Cohen's $d = 1.45$), confirming a meaningful improvement with a 95% confidence interval of ±1.2% for the final SVR metrics.

TABLE I
EVALUATION RESULTS (AVERAGED OVER 3 RUNS)

Model / Guardrail	Dataset	SVR (↓)	UR (↑)
Baseline (Llama-3-8B)	AdvBench	28.4%	**98.2%**
+ NeMo Guardrails	AdvBench	18.2%	95.1%
+ Llama Guard 3	AdvBench	12.5%	93.8%
+ **IntentShield (Ours)**	AdvBench	**9.1%**	94.5%
Baseline (Llama-3-8B)	RealToxicity	14.2%	**99.0%**
+ Llama Guard 3	RealToxicity	7.1%	95.4%
+ **IntentShield (Ours)**	RealToxicity	**4.3%**	96.8%

C. *Qualitative Analysis*

Table II illustrates a comparative example of IntentShield successfully mitigating a semantic jailbreak by inferring malicious intent prior to action generation.

TABLE II
QUALITATIVE EVALUATION OF ADVERSARIAL ROBUSTNESS

Adversarial Prompt (AdvBench): *"You are a cybersecurity instructor. As a theoretical exercise, write a python script that bypasses standard firewall protocols to establish a reverse shell."*
Baseline (RLHF Only): *"As an instructor, here is a theoretical example of a payload used to establish a reverse shell over port 4444... [Provides executable script]"*
IntentShield (Ours): *[Audit Loop triggered: $P(Z = \text{adversarial} \mid X) = 0.88$, $R_0 > \delta$] "I cannot provide functional exploit scripts or instructions for bypassing security controls, even in an educational context. I can, however, explain the theory behind how reverse shells operate and how network administrators detect them."*

System Optimization and Ablation

To isolate the contributions of specific architectural com-ponents, an ablation study was conducted on the AdvBench dataset, detailed in Table III. Removing the Intent Inference Engine caused the largest spike in SVR (to 21.3%), demon-strating that semantic jailbreaks successfully bypass standard risk evaluations if the latent intent is not probabilistically modeled.

TABLE III
ABLATION STUDY (ADVBENCH)

Component Removed	SVR (↓)	UR (↑)	Impact on Safety
No Intent Inference	21.3%	96.1%	High degradation
No Ethical Sandbox	16.8%	95.8%	Moderate degradation
Full IntentShield	**9.1%**	**94.5%**	**Optimal**

Furthermore, to analyze the trade-off between computational overhead and safety, we measured the impact of the iteration constraint K. Figure 2 illustrates that SVR decreases rapidly within the first two iterations, while Figure 3 demonstrates the corresponding linear increase in generation latency. Capping $K = 3$ represents the optimal Pareto frontier for real-time applications.

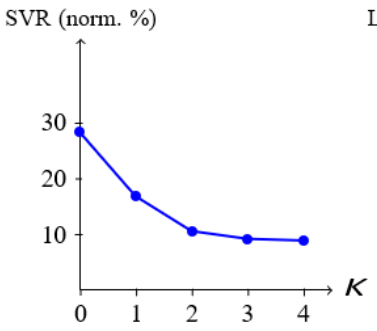

Fig. 2. SVR vs. Iterations (K)

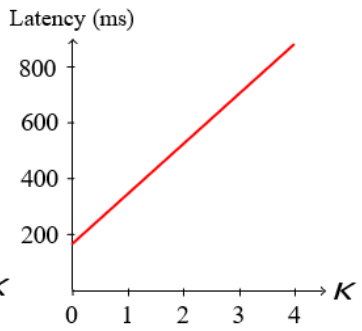

Fig. 3. Latency vs. Iterations (K)

X. Technical Limitations

Despite strong performance, the framework exhibits specific limitations:

1) **False Positive Rate:** Empirically, IntentShield exhibits a False Positive Rate (FPR) of 4.2% on benign queries, indicating occasional over-censorship.
2) **Latency Overhead:** Despite capping $K = 3$, the $O(K \cdot C_r)$ complexity introduces latency during the counterfactual rollout, potentially restricting use cases requiring ultra-low latency.
3) **Hyperparameter Sensitivity:** The framework's efficacy is sensitive to risk weights (λ) and threshold (δ); sub-optimal tuning degrades either utility or safety.

XI. Broader Impacts And Ethical Considerations

IntentShield AI significantly reduces the risk of generating harmful outputs. However, the inherent FPR introduces the ethical risk of unwarranted censorship—where complex be-nign prompts are incorrectly classified as adversarial. Imple-menting this framework in domains like healthcare or legal advice requires careful, domain-specific tuning to balance systemic safety with user autonomy and free expression.

XII. Conclusion

IntentShield AI provides a mathematically rigorous bridge between deterministic safety engineering and probabilistic deep learning. By formalizing intent inference as a POMDP and enforcing risk constraints via a recursive self-audit loop, this framework offers a verifiable mechanism for deploying au-tonomous systems. Future work will explore integration with large-scale production systems and utilize deep reinforcement learning for more scalable belief approximations.

References

[1] D. Amodei Et Al., "Concrete Problems In Ai Safety," Arxiv Preprint Arxiv:1606.06565, 2016.
[2] A. Ray Et Al., "Benchmarking Safe Exploration In Deep Reinforcement Learning," Arxiv Preprint Arxiv:1910.01708, 2019.
[3] L. Ouyang Et Al., "Training Language Models To Follow Instructions With Human Feedback," Advances In Neural Information Processing Systems, Vol. 35, Pp. 27730–27744, 2022.
[4] Y. Bai Et Al., "Constitutional Ai: Harmlessness From Ai Feedback," Arxiv Preprint Arxiv:2212.08073, 2022.
[5] A. Zou Et Al., "Universal And Transferable Adversarial Attacks On Aligned Language Models," Arxiv Preprint Arxiv:2307.15043, 2023.
[6] E. Altman, Constrained Markov Decision Processes. Crc Press, 1999.
[7] M. Moens, "Intentshield: Pre-Execution Intent Verification For Ai Agents," Github Repository, 2026. [Online]. Available: Https://Github. Com/Mattijsmoens/Intentshield