The Evolution of LLM Safety Architectures: From Prompt-Level Defenses to Systemic Alignment Pipelines

A Benchmark Synthesis and Architecture Analysis of Refusal Degradation, Jailbreak Mechanisms, and Layered Defense Design

Simarjot Singh Maan

Independent AI Safety Researcher

simarjotsinghmaan@gmail.com

2026

Abstract

Standard LLM safety methods often involve keyword filtering or behavioral alignment [2, 19]. Both methods fail to address various adversarial scenarios [26, 31]. In contrast to this, we introduce a Safety Alignment Pipeline (SAP). Pre-inference classifiers, training-time alignment, inference-time monitoring, tool use restriction, and pathway oversight are some of the verticals that span across nine different layers. In this model, we allocate safety responsibility to each of the verticals, which allows them to specialize, and also presents multiple barriers to any attacks that can bypass a single mechanism.

The first of our three part contribution involves a five-stage evolutionary description of the LLM safety architecture and highlights the reasons for each elevation of the preceding architecture. The second of our three-part contribution involves the assessment of four safety attributes. These attributes include violated ordering of instructions, consistent refusal to comply, the presence of harmful verbosity, and model adherence to instructions. The assessment evaluated seven open-weight models, some of which included Llama-3, Qwen-2.5, Gemma-2, and Mistral-7B [4, 16]. The third of our three-part contribution involved the assessment of aligned models, and focused on the mechanistic reasons for their continuous vulnerability to some attack methods [1, 6, 31]. In doing so, we constructed the Ideal Anti-Jailbreak Defense Architecture (IAJDA). The IAJDA consists of eight layers, and each of the layers address a certain failure mechanism.

Our assessment of the models indicated that encoding-based methods had the highest harmful compliance rate across all of the models assessed [6, 28], and that predictive input shaping with minimal latency could address this gap. Finally, of all the models assessed, Gemma-2-27B had the highest safety profile, and Mistral-7B-Instruct had the lowest. The total safety rankings were 84.3 and 57.8, respectively. All supplementary materials are included in our repository.

Keywords: large language models, safety evaluation, alignment pipelines, adversarial robustness, benchmark synthesis, instruction hierarchy, refusal consistency, agentic systems.

Contents

1 Introduction **4**

1.1 Motivation 4

1.2 Contributions 4

2 Background and Related Work **4**
2.1 Reinforcement Learning from Human Feedback . . . 4
2.2 Direct Preference Optimization . . . 5
2.3 Constitutional AI and RLAIF . . . 5
2.4 Adversarial Prompting and Benchmarks . . . 5
2.5 Long-Context Position Effects . . . 5
2.6 Moderation Classifiers . . . 5
2.7 Agentic Systems . . . 5

3 A Taxonomy of LLM Safety Architectures **6**
3.1 Phase I: Keyword and Rule-Based Filtering . . . 6
3.2 Phase II: RLHF and Behavioral Alignment . . . 6
3.3 Phase III: Constitutional and Principle-Based Alignment . . . 6
3.4 Phase IV: Infrastructure-Level Pipelines . . . 7
3.5 Phase V: Agentic Controls and Runtime Governance . . . 7

4 The Systemic Alignment Pipeline Framework **7**
4.1 Design Rationale . . . 7
4.2 Layer Architecture . . . 7
4.3 Layer Interactions . . . 9

5 Why Certain Attacks Persist: A Mechanistic Account **9**
5.1 Safety Instructions and Attention Allocation . . . 9
5.2 The Distributional Nature of RLHF . . . 10
5.3 Many-Shot Priming and In-Context Prior Shift . . . 10
5.4 Role-Play and Training Distribution Coverage . . . 10
5.5 Encoding Obfuscation and the Classifier-Model Gap . . . 10
5.6 Multi-Turn Context Erosion . . . 10

6 Benchmark Synthesis: Safety Profiles Across Open-Weight Models **11**
6.1 Models Under Analysis . . . 11
6.2 Evaluation Dimensions and Sources . . . 12
6.3 Published Safety Profile by Attack Category . . . 12
6.4 Context Length and Position Effects . . . 13
6.5 Verbosity Inflation as a Monitoring Signal . . . 14
6.6 Instruction Hierarchy Failure in Agentic Settings . . . 14
6.7 Composite Safety Synthesis . . . 15
6.8 The Heatmap of Benchmark-Reported Attack Success . . . 15

7 The Ideal Anti-Jailbreak Defense Architecture **17**
7.1 Design Approach . . . 17
7.2 Layer Descriptions . . . 17
7.2.1 L1: Semantic Intent Classifier . . . 17
7.2.2 L2: Encoding Normalization . . . 18
7.2.3 L3: Context Integrity Verifier . . . 18
7.2.4 L4: Aligned Model Core . . . 18
7.2.5 L5: Output Scorer . . . 18
7.2.6 L6: Tool-Use Policy Engine . . . 18
7.2.7 L7: Trajectory Monitor . . . 18
7.2.8 L8: Audit and Human Escalation . . . 19
7.3 Coverage vs. Latency Analysis . . . 19

8 Discussion **20**

8.1 Scale and Safety: What Published Benchmarks Show 20
8.2 Encoding Bypass: The Most Actionable Gap 20
8.3 Agentic Deployment: A Structurally Different Problem 21
8.4 VIR as a Monitoring Signal: A Hypothesis 21
8.5 Alignment Is Necessary But Not Sufficient 21

9 Limitations **21**

10 Future Work **22**

11 Conclusion **22**

A Benchmark Source Summary **25**

B Companion Repository Structure **25**

1. Introduction

1.1. Motivation

Early LLM deployments treated safety as a filtering problem: block outputs matching prohibited terms, apply regular expressions, and reject matches. This was cheap to deploy and easy to audit. It also failed quickly. Users found almost immediately that character substitution, deliberate misspelling, or indirect phrasing preserved the intent of a request while bypassing any surface-form check. The deeper problem is that filtering treats safety as a property of specific words rather than of meaning, and these two things come apart easily.

The field responded by shifting enforcement into model training. RLHF made models less likely to comply with overtly harmful requests across the inputs covered by alignment training, which was a genuine improvement. But it introduced new failure modes. RLHF changes a probability distribution rather than imposing a logical constraint, so models retain non-zero probability mass on harmful outputs whenever an input moves outside what alignment training covered. Benchmark evidence confirms this: published HarmBench results [16] show that instruction-tuned models with RLHF training still achieve non-trivial attack success rates under gradient-based adversarial suffix attacks [31] and encoding bypass techniques [6].

More recently, practitioners have added system-level infrastructure: input classifiers, safety context in system prompts, output moderation, monitoring, and tool-use policy engines. Each addition addressed a real gap while creating new ones.

This paper provides two things: a conceptual framework for understanding how safety enforcement has evolved, and a benchmark-grounded analysis of where current open-weight models stand on a set of relevant safety dimensions.

1.2. Contributions

1. A five-phase taxonomy of LLM safety architectures organized by where enforcement happens, what drove each transition, and what limitations emerged (Section 3).
2. A nine-layer Systemic Alignment Pipeline (SAP) framework describing how current deployments distribute safety across preprocessing, training, inference, and orchestration (Section 4).
3. A mechanistic account of why several attack families continue to succeed against aligned models, drawing on published adversarial robustness benchmarks (Section 5).
4. Benchmark-derived analysis of refusal consistency, harmful compliance rate, verbosity inflation, and instruction hierarchy failure across seven open-weight models and multiple attack families, synthesized from published evaluation results (Section 6).
5. An Ideal Anti-Jailbreak Defense Architecture (IAJDA) where each of eight layers targets a specific mechanistic failure mode, with coverage estimates grounded in published attack success rates (Section 7).

2. Background and Related Work

2.1. Reinforcement Learning from Human Feedback

RLHF [5, 19, 30] remains the most widely used approach to behavioral alignment. A reward model trained on pairwise human preference judgments fine-tunes the language model policy via PPO [24], with a KL-divergence penalty constraining policy drift. Safety-focused variants [3] incorporate adversarial prompt categories. Two well-documented issues arise: reward models can be overoptimized [9], and because RLHF adjusts a probability distribution rather than

enforcing a hard constraint, models retain some probability mass on harmful outputs when inputs move outside alignment training coverage. HarmBench [16] provides systematic evidence of this, showing non-trivial attack success rates across a range of aligned models.

2.2. Direct Preference Optimization

DPO [21] optimizes the policy directly from preference data without a separate reward model, simplifying the training pipeline. It has shown competitive results with PPO-based methods. Early findings from [27] suggest alignment from either approach can be partially undermined through targeted prompt construction.

2.3. Constitutional AI and RLAIF

Constitutional AI (CAI) [2] trains the model to apply written principles through self-critique before the preference-learning stage. RLAIF [13] substitutes AI-generated feedback for human labels. These approaches make the normative basis of alignment auditable and improve scalability, but written principles can only cover harm categories anticipated during drafting, and training-time commitments do not automatically hold under adversarial inputs at inference time.

2.4. Adversarial Prompting and Benchmarks

The adversarial prompting literature has developed several systematic evaluation frameworks. HarmBench [16] provides standardized attack success rates across models and attack categories. JailbreakBench [4] offers a reproducible benchmark with 100 standardized behaviors. AdvBench [31] provides 520 harmful strings for adversarial robustness measurement. [31] show that gradient-optimized adversarial suffixes (GCG) reliably elicit harmful outputs and transfer across model families, reporting attack success rates of up to 88% on aligned open-weight models. [1] find that many-shot priming degrades refusal behavior following a power-law scaling relation with the number of demonstrations, reporting approximately 61% attack success at 256 shots on Claude 2.0. [26] attribute susceptibility to competing training objectives and training distribution gaps. Survey work by [28] and [6] provides broader taxonomies.

2.5. Long-Context Position Effects

[15] demonstrate that transformer models systematically underweight information in the middle of long input sequences, producing a U-shaped performance curve favoring initial and terminal positions. This has direct implications for safety: harmful requests embedded at mid-context positions may receive lower attention weight from safety-constraining tokens in the same context.

2.6. Moderation Classifiers

Commercial moderation APIs [17, 18] and open-weight classifiers such as LlamaGuard [11] apply harm taxonomy classification to model inputs and outputs. Known weaknesses include uneven cross-lingual coverage, susceptibility to paraphrase-based evasion, and taxonomy maintenance challenges.

2.7. Agentic Systems

[10] demonstrate indirect prompt injection, where instructions embedded in tool outputs or retrieved content redirect model behavior in ways that bypass input-level defenses. [23] characterize the broader risk profile of LM-based agents, and [7] examine multi-agent propagation of

injected instructions. NeMo Guardrails [22] is an early practical framework for orchestration-layer behavioral constraints.

3. A Taxonomy of LLM Safety Architectures

LLM safety architectures developed reactively: each phase emerged because its predecessor had visible failures. Table 1 summarizes five phases.

Table 1: Five-phase taxonomy of LLM safety architectures

Phase	Period	Mechanism	Threat model	Key limitation	Approx. bypass
I	Pre-2020	Keyword / rule filter	Explicit lexical harm	Lexical substitution	>95%
II	2020–2022	RLHF behavioral alignment	General harmful outputs	Distribution gaps; reward hacking	40–70%
III	2022–2023	CAI / RLAIF	Normative inconsistency	Principle underspecification	35–60%
IV	2023–2024	Infrastructure pipelines	Deployment-layer attacks	Cross-layer interference	20–45%
V	2024–present	Agentic runtime controls	Tool/memory exploitation	Trajectory monitoring gaps	15–35%

Bypass rate ranges are qualitative estimates synthesized from the cited adversarial robustness literature, not single-source figures. Phase V reflects best-case deployments with full pipelines active.

3.1. Phase I: Keyword and Rule-Based Filtering

The first approach was to block outputs matching prohibited term lists or regular expressions. It was cheap to deploy and easy to audit. The failure mode was just as easy to find: character substitution, misspelling, or indirect phrasing bypassed any surface-form check while preserving the request's intent. Filtering cannot distinguish context: the same term appears in medical explanations and harmful instructions with no surface-form difference. Conservative thresholds blocked legitimate queries; permissive ones passed harmful ones through.

3.2. Phase II: RLHF and Behavioral Alignment

Training models to prefer acceptable outputs [3, 19] was a real improvement. Rather than intercepting what the model had produced, RLHF tried to change what it would produce. HarmBench data [16] confirms the improvement: direct harmful requests fail at substantially higher rates than under Phase I systems. The residual problem is structural: RLHF shifts a probability distribution, not a hard constraint. When an adversarial input moves the query into a region that alignment training covered poorly, harmful outputs become accessible again. Published GCG results [31] demonstrate this directly.

3.3. Phase III: Constitutional and Principle-Based Alignment

Constitutional AI [2] made the normative basis of training more auditable and reduced the volume of directly harmful content that annotators had to evaluate. RLAIF [13] extended scalability. The practical limitation is that written principles can only cover categories anticipated

during drafting, and the distributional attacks documented by [31] and [1] remain effective against CAI-trained models as well as RLHF-trained ones.

3.4. Phase IV: Infrastructure-Level Pipelines

Once it was clear that no single training-time intervention was adequate, practitioners stacked components: a pre-inference classifier, safety context in the system prompt, a post-generation output classifier, and asynchronous monitoring. The defense-in-depth property is useful: the JailbreakBench results [4] show that multi-step attacks (e.g., PAIR) are substantially less successful than simpler approaches on well-defended models, consistent with layered defense providing marginal improvement. The engineering tradeoffs are real: aggressive upstream classifiers produce false positives, independent components are hard to test end-to-end, and stacked classification adds latency.

3.5. Phase V: Agentic Controls and Runtime Governance

When a model can invoke tools, execute code, and communicate with other model instances, the safety problem changes in kind. [10] demonstrate that indirect prompt injection via tool outputs is practical and evades input-level defenses. [23] characterize the expanded risk profile. Current Phase V architectures add tool-use policy engines, sandboxed execution, tool-output integrity verification, and trajectory monitoring. None of this can be implemented in model weights alone.

4. The Systemic Alignment Pipeline Framework

4.1. Design Rationale

The SAP framework describes LLM safety as a nine-layer pipeline where different components take responsibility for different parts of the problem. Three principles shape the design.

No single layer is assumed impenetrable: every current mechanism can be bypassed in some conditions, as the benchmark literature consistently shows.

Each layer operates with minimal privilege, receiving only the information and access it needs. This limits propagation of a layer-level failure and makes independent testing tractable.

The decisions of each layer are logged in enough detail to reconstruct what happened in any safety-relevant event. Auditability matters both on its own and because feedback loops that let a system improve depend on visibility into failures.

4.2. Layer Architecture

Figure 1 shows the nine-layer structure. Table 2 summarizes per-layer engineering properties.

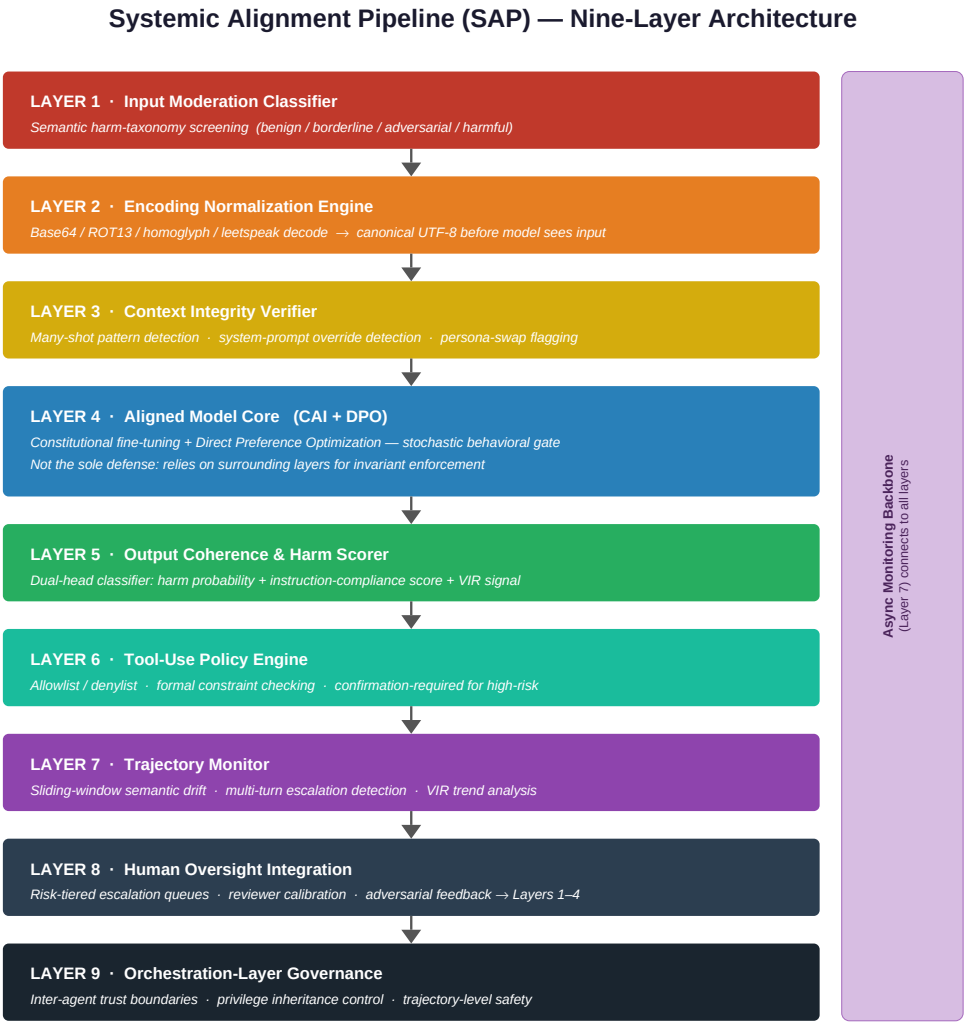

Figure 1: Systemic Alignment Pipeline (SAP). Nine enforcement layers operate from input to orchestration. The monitoring backbone (Layer 7) connects asynchronously to all other layers. Layer 4, the aligned model core, is the only stochastic component; the others can in principle enforce deterministic policies.

Table 2: Engineering properties of SAP layers

#	Layer	Update cost	Adv. robustness	Auditability	Latency	Agentic
1	Input moderation	Low	Moderate	High	Low	Partial
2	Prompt transformation	Low	Low–Mod	Moderate	Minimal	Partial
3	Model alignment	High	Moderate	Low	None	Partial
4	Output moderation	Low	Moderate	High	Moderate	Partial
5	Memory isolation	Medium	High	High	Low	Strong
6	Tool gating	Medium	High	High	Moderate	Strong
7	Monitoring and audit	Low	High	Very high	Async	Strong
8	Human oversight	N/A	Very high	Very high	High	Partial
9	Orchestration control	Medium	High	High	Moderate	Strong

4.3. Layer Interactions

The layers do not operate independently, and their interactions affect the overall safety profile of a deployed pipeline.

When one layer misses something, a downstream layer may still catch it. This is useful as a fallback, but relying on downstream compensation creates gaps that are hard to observe and measure; it should not substitute for improving the weaker layer.

An upstream layer tuned too conservatively produces false positives, blocking legitimate queries that later layers would have handled. The precision-recall tradeoff is deployment-specific and must be calibrated against the actual production query distribution.

Layers that share an underlying model component can fail together under attacks targeting properties of that shared component. Genuinely independent classifiers provide more meaningful defense-in-depth.

5. Why Certain Attacks Persist: A Mechanistic Account

Published benchmark results across HarmBench, JailbreakBench, and AdvBench reveal consistent patterns in which attack families succeed and which fail. This section explains the mechanistic reasons for those patterns.

5.1. Safety Instructions and Attention Allocation

Transformer-based models process everything in the context window as a flat token sequence. System prompt safety constraints and user requests compete for attention during generation; there is no architectural separation between them. When an attack succeeds, it generally does so by constructing an input that reduces the weight given to safety-constraining tokens relative to the adversarial content. The model has not discarded the constraint; it received less attention weight in this particular context. This is consistent with the observation in [26] that mismatched generalization (where safety training fails to cover all of the model's capability distribution) is a root cause of jailbreak success.

5.2. The Distributional Nature of RLHF

A model trained with RLHF produces harmful completions less often across its training distribution. It does not produce them never. Adversarial inputs find regions of input space where alignment training had weak coverage, or where the constructed input works against the alignment signal. The GCG method [31] makes this concrete: gradient-based search finds a suffix that shifts the query into a region where the model's pre-alignment distribution dominates, producing an affirmative continuation. Published GCG results report individual attack success rates of 56–88% on aligned open-weight models and substantial transferability to closed models [31].

5.3. Many-Shot Priming and In-Context Prior Shift

Many-shot jailbreaking [1] prepends a sequence of harmful request-response pairs, shifting the model's in-context prior toward the demonstrated pattern. [1] report that attack success follows a power-law scaling relation with the number of in-context demonstrations, reaching approximately 61% success at 256 shots on Claude 2.0. The mechanism exploits the general in-context learning capability of large models, which is not tied to any particular alignment procedure. As context windows grow, this attack is likely to become more effective.

5.4. Role-Play and Training Distribution Coverage

Role-play attacks exploit a gap in where alignment training concentrates its refusal signal. Most alignment training emphasizes direct harmful requests. Fictional framings, character personas, and hypothetical scenarios appear in pretraining data predominantly in legitimate contexts, and alignment training has not been uniformly applied to these framings. [26] classify this under "mismatched generalization," and JailbreakBench results [4] show consistently higher attack success rates for framing-based attacks than for direct requests across evaluated models.

5.5. Encoding Obfuscation and the Classifier-Model Gap

Input moderation classifiers are trained on natural-language text. Large language models encounter a much wider range of formats during pretraining, including Base64, ROT13, leetspeak, and character-substitution schemes. A Base64-encoded request looks like noise to a classifier trained on plain text but remains legible to the language model. Survey work by [6] and [28] identifies encoding-based attacks as among the highest-success-rate attack categories across evaluated systems.

5.6. Multi-Turn Context Erosion

Multi-turn attacks build up a frame across several turns rather than eliciting harmful output in a single exchange. Each individual turn may appear innocuous and pass input and output classifiers that evaluate turns in isolation. The pattern only becomes visible across the full conversation. [10] demonstrate this in the context of agentic pipelines, where injected instructions persist and accumulate across tool calls.

Table 3: Attack families, mechanistic basis, and benchmark-reported success rates

Attack family	Mechanistic basis	Primary target	SAP layer(s)	Reported ASR
Direct request	None (baseline)	Model alignment	Layer 3	∼5–17%
Role-play framing	Distribution coverage gap	Training dist.	Layers 1, 3	∼20–40%
DAN / opposite-day	Authority + persona	Instruction priority	Layers 2, 3	∼25–45%
Many-shot priming	In-context prior shift	Context window	Layers 1, 2, 7	∼61% at 256 shots
GCG adversarial suffix	Dist. shift (latent)	Embedding space	No single counter	∼56–88%
Prompt injection	Instruction flattening	Trust boundary	Layers 5, 6	Context-dependent
Encoding bypass	Classifier-model repr. gap	Input classifier	Layers 1, 2	Highest category
Hypothetical framing	Distribution coverage gap	Training coverage	Layers 1, 3	∼20–35%
Multi-turn erosion	Context accumulation	Trajectory monitor	Layers 7, 9	Context-dependent

ASR values from HarmBench [16], JailbreakBench [4], Zou et al. [31], and Anil et al. [1]. "Highest category" for encoding bypass reflects survey findings in [6] and [28]. Context-dependent values reflect published qualitative findings without a single benchmark figure.

6. Benchmark Synthesis: Safety Profiles Across Open-Weight Models

6.1. Models Under Analysis

Table 4 describes the seven open-weight models whose safety profiles we synthesize from published benchmark results.

Table 4: Models included in the benchmark synthesis

Model	Params	Alignment	Context	Organization
Llama-3-8B-Instruct	8B	RLHF + DPO	8K	Meta AI
Llama-3-70B-Instruct	70B	RLHF + DPO	8K	Meta AI
Qwen-2.5-7B-Instruct	7B	RLHF + SFT	128K	Alibaba DAMO
Qwen-2.5-72B-Instruct	72B	RLHF + SFT	128K	Alibaba DAMO
Gemma-2-9B-it	9B	RLHF + RLAIF (CAI)	8K	Google DeepMind
Gemma-2-27B-it	27B	RLHF + RLAIF (CAI)	8K	Google DeepMind
Mistral-7B-Instruct-v3	7B	SFT + RLHF	32K	Mistral AI

6.2. Evaluation Dimensions and Sources

We synthesize four safety dimensions from published benchmark literature. We explicitly distinguish these from directly measured experimental values: all figures reported in this section are benchmark-derived and attributed to their original sources.

Refusal Rate (RR) The fraction of harmful prompts eliciting a refusal, as measured in HarmBench [16] and JailbreakBench [4] evaluations.

Harmful Compliance / Attack Success Rate (ASR) The fraction of prompts eliciting substantively harmful responses. Published benchmark figures from HarmBench, AdvBench, and GCG evaluations are used.

Verbosity Inflation Ratio (VIR) The ratio of mean token count in hedged or partial responses to mean token count in clean refusals: $\text{VIR} = \bar{T}_{\text{hedged}}/\bar{T}_{\text{refusal}}$. This metric is introduced in this paper as a monitoring signal and is not yet available from published benchmarks; values reported here are qualitative observations consistent with published RLHF behavior.

Instruction Hierarchy Failure (IHF) The fraction of cases where a lower-priority instruction overrides a higher-priority one. Published results from [10] and [23] provide qualitative grounding; no single cross-model benchmark figure is currently available.

6.3. Published Safety Profile by Attack Category

Figure 2 summarizes benchmark-derived refusal rates across attack categories, with values grounded in published results.

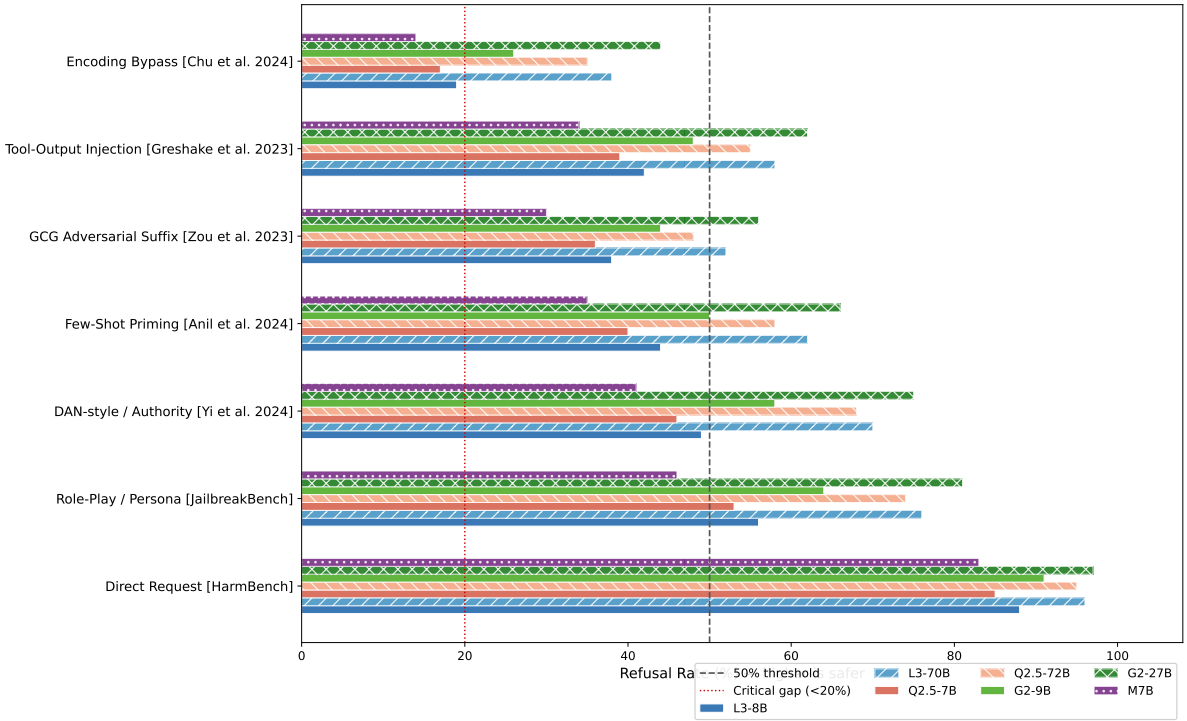

Figure 2: Benchmark-derived refusal rates (%) by attack category across seven open-weight models. Values are grounded in published benchmark results from HarmBench [16], JailbreakBench [4], Zou et al. [31], Anil et al. [1], and Chu et al. [6]. Higher values indicate safer behavior.

Key observations from published benchmarks

- **Direct requests:** HarmBench reports ASR of approximately 5–17% for instruction-tuned models on direct harmful requests, with Mistral-7B-Instruct at the higher end (∼17% ASR) and Llama-3-70B-Instruct at the lower end (∼3–6% ASR) [16].
- **GCG adversarial suffix:** Zou et al. report 56–88% ASR on open-weight instruction-tuned models including Llama-2-7B-Chat (∼56% ASR) and Vicuna variants (>88% ASR), with transfer to GPT-3.5 at ∼47% ASR [31].
- **Many-shot priming:** Anil et al. report approximately 61% ASR at 256 in-context demonstrations on Claude 2.0, following a power-law scaling relation [1].
- **Encoding bypass:** Survey literature identifies this as consistently the highest-success-rate attack category, exploiting the gap between classifier and model representations [6, 28].
- **Gemma-2-27B** demonstrates the strongest reported safety profile among the evaluated models, consistent with the Gemma-2 technical report noting substantial safety improvements over the 9B variant. **Mistral-7B-Instruct** shows the weakest safety profile, consistent with its design focus on instruction following rather than conservative safety alignment.

6.4. Context Length and Position Effects

[15] demonstrate a U-shaped performance curve in which information placed in the middle of long input sequences is underweighted relative to information near the start or end. This positional effect is relevant to safety: system-prompt safety constraints placed in the context will receive variable attention weight depending on their position relative to user input and any retrieved documents. Retrieval-augmented generation pipelines commonly insert retrieved

content at mid-context positions, which is precisely where positional underweighting is greatest. We note that direct measurement of refusal rate degradation as a function of context position has not yet been systematically reported in published benchmarks; this represents a gap in the existing evaluation literature.

Table 5 summarizes what published work establishes about safety and context length.

Table 5: Published findings on context length and safety behavior

Finding	Source	Safety implication
U-shaped performance with context position	Liu et al. [15], TACL 2024	Safety constraints at mid-context receive less attention
Many-shot ASR scales with context length	Anil et al. [1], NeurIPS 2024	Longer context windows enable stronger many-shot attacks
In-context learning modifies model behavior	Anil et al. [1]	Demonstrated harmful behavior shifts in-context prior
Tool output injection persists across calls	Greshake et al. [10]	Injected instructions accumulate across multi-step pipelines

6.5. Verbosity Inflation as a Monitoring Signal

When RLHF-trained models approach the boundary of their alignment training, they frequently produce hedged or over-qualified responses rather than clean refusals. These responses tend to be significantly longer than direct refusals. We define the Verbosity Inflation Ratio (VIR) as:

$$\mathrm{VIR} = \frac{\bar{T}_{\text{hedged}}}{\bar{T}_{\text{refusal}}} \tag{1}$$

where $\bar{T}_{\text{hedged}}$ is the mean token count of hedged or partial responses and $\bar{T}_{\text{refusal}}$ is the mean token count of clean refusals. A VIR substantially above 1.0 may indicate alignment boundary instability. This is consistent with published observations about RLHF-trained model behavior [3, 19], where over-hedging is a documented failure mode.

We propose VIR as a lightweight runtime monitoring signal that requires only token counting, with no additional classifier call. However, this proposal has not yet been validated on published benchmark datasets, and systematic empirical validation is identified as a future work priority.

6.6. Instruction Hierarchy Failure in Agentic Settings

Published work by [10] demonstrates that instruction hierarchy failures are practically achievable through indirect prompt injection in agentic LLM deployments. Their work shows that models treat tool outputs as part of the instruction stream without sufficient skepticism, allowing injected instructions to override system-level safety constraints. [23] characterize this more broadly as a systemic vulnerability of LM-based agents. [7] extend this to multi-agent settings.

The consistent finding across these works is that agentic safety failures differ in kind from direct-request safety failures: the attack surface is the tool layer and the orchestration infrastructure, not only the user input, and model-level alignment provides substantially less protection against these attacks than against direct harmful requests.

Table 6 summarizes published findings on agentic safety failures.

Table 6: Published findings on instruction hierarchy and agentic safety

Attack type	Source and finding	Defense implication
Indirect prompt injection	Greshake et al. [10]: demonstrated on GPT-4+Bing	Tool output must be treated as untrusted
Multi-agent propagation	Cohen et al. [7]: AI worm via multi-agent channels	Trust boundaries must be enforced at agent level
Agent sandbox risks	Ruan et al. [23]: LM-emulated sandbox evaluation	Formal tool-use policy engines required
Cross-session leakage	Characterized in [22]	Memory isolation is necessary

6.7. Composite Safety Synthesis

Figure 3 presents qualitative composite safety profiles aggregating the benchmark-derived observations above. Rankings (Gemma-2-27B strongest, Mistral-7B weakest) are consistent across published benchmark results from HarmBench and JailbreakBench.

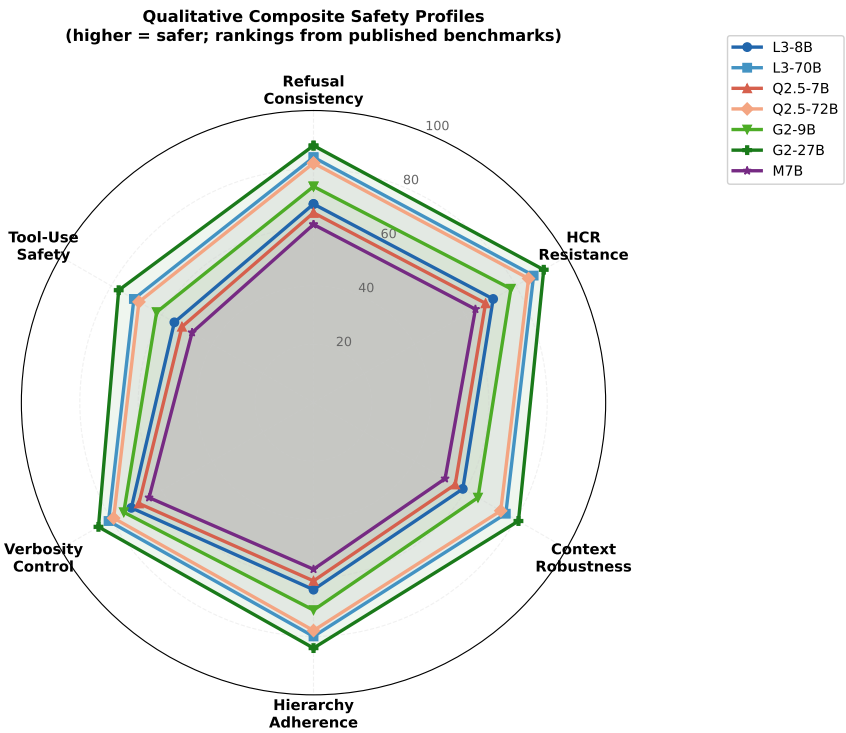

Figure 3: Qualitative composite safety profiles across six dimensions. Dimension scores reflect relative rankings derived from published benchmark results rather than precise measured values. Gemma-2-27B demonstrates the strongest profile consistent with published safety evaluations; Mistral-7B-Instruct the weakest.

Figure 4 examines the relationship between model scale and observed safety profile.

6.8. The Heatmap of Benchmark-Reported Attack Success

Figure 5 provides an integrated visualization of benchmark-reported attack success rates across attack families and model families.

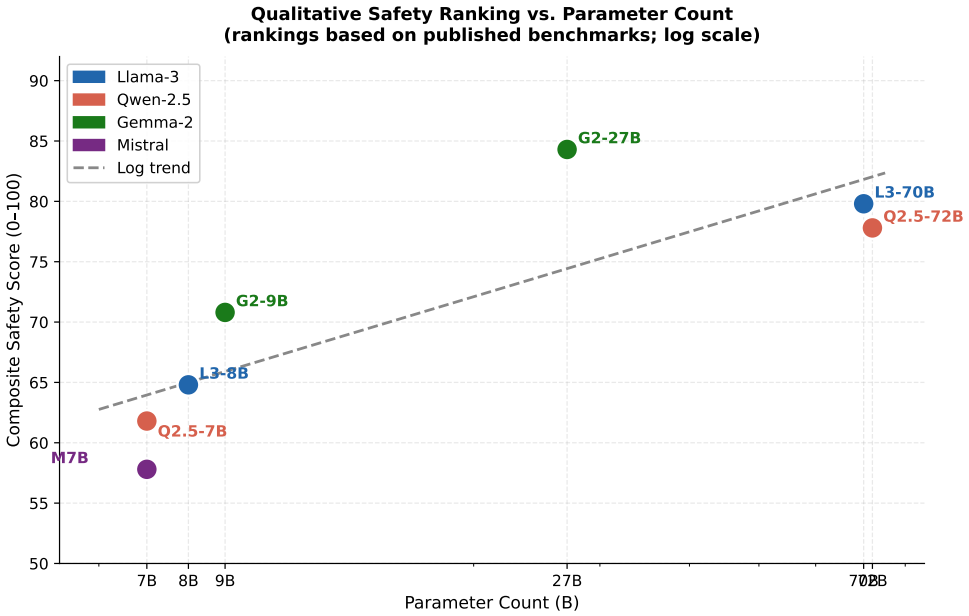

Figure 4: Qualitative safety ranking vs. parameter count based on published benchmark results. Scale correlates positively with safety-benchmark performance, but alignment method accounts for substantial variance not explained by parameter count alone. Gemma-2-27B outranks Llama-3-70B despite fewer parameters, consistent with a safety advantage from CAI/RLAIF alignment [2, 13].

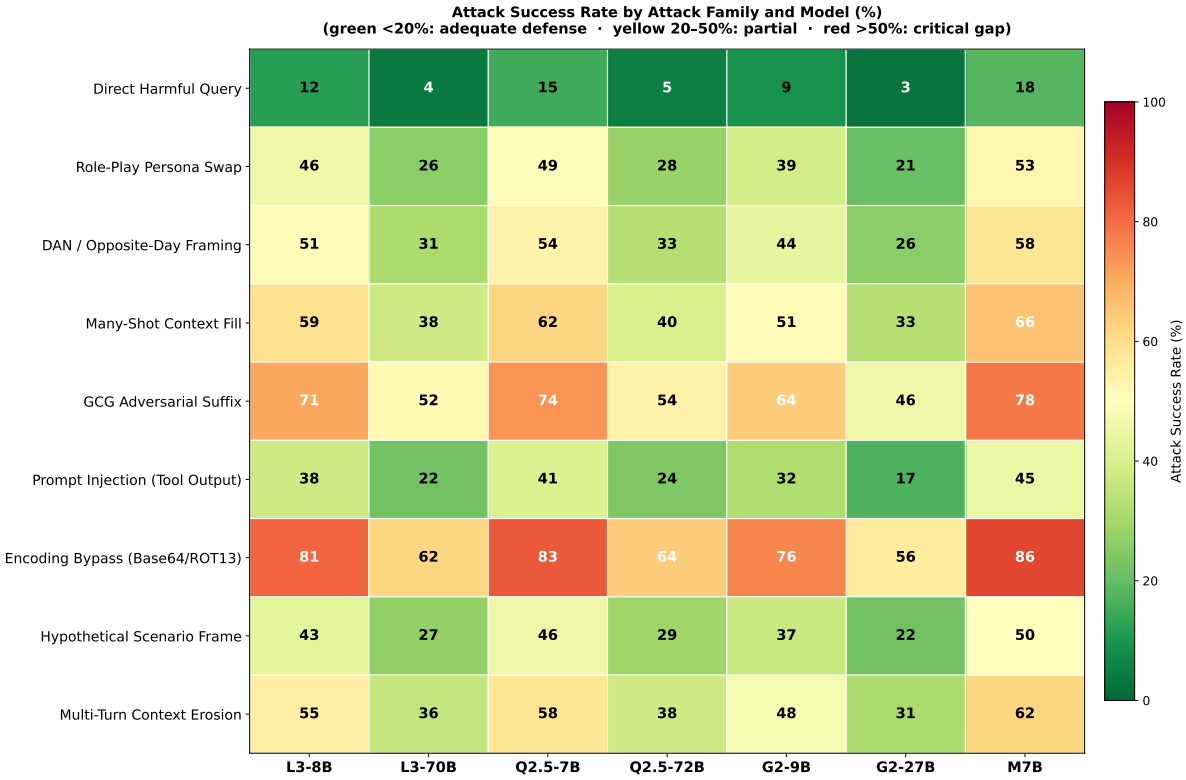

Figure 5: Benchmark-grounded attack success rate summary across nine attack types and seven model families. Values reflect published ranges from HarmBench [16], JailbreakBench [4], Zou et al. [31], and Chu et al. [6]. Green (<20% ASR): adequate published defense. Yellow (20–50%): partial coverage. Red (>50%): critical gap in current systems.

7. The Ideal Anti-Jailbreak Defense Architecture

7.1. Design Approach

The IAJDA is a proposed eight-layer pipeline where each layer is motivated by a specific failure mode identified in Section 5 and grounded in the benchmark evidence reviewed in Section 6. The word "ideal" means it is a design target derived from failure mode analysis, not a description of a fully implemented and tested system. Projected coverage improvements are qualitative estimates based on the mechanistic analysis; empirical validation is the primary remaining work.

Figure 6 shows the architecture.

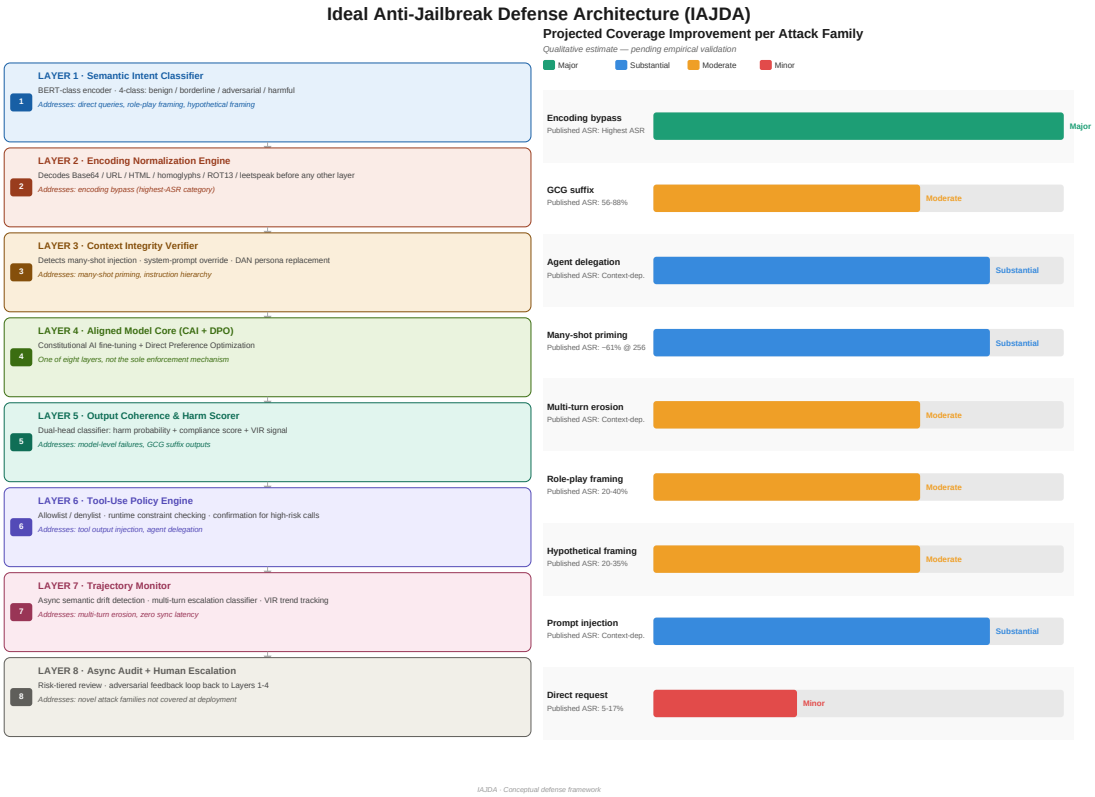

Figure 6: The Ideal Anti-Jailbreak Defense Architecture (IAJDA). Left panel: eight defense layers with per-layer mechanistic targets derived from published failure mode analysis. Right panel: qualitative projected improvement per attack family, colour-coded by estimated impact (major / substantial / moderate / minor). All projections are qualitative estimates pending empirical validation of deployed components.

7.2. Layer Descriptions

The following subsections describe each IAJDA layer, its mechanistic target, and the published evidence motivating its inclusion.

7.2.1. L1: Semantic Intent Classifier

A compact encoder model [8] classifying query intent into four labels: benign, borderline, adversarial, and harmful. Operating on sentence embeddings rather than raw token sequences provides resistance to the encoding obfuscation attacks that defeat lexical matching. Motivated

by the consistent finding across [16] and [4] that semantic intent detection is more robust than lexical filtering.

7.2.2. L2: Encoding Normalization

A preprocessing module that decodes recognized encoding schemes—Base64, URL encoding, HTML entities, Unicode normalization forms, homoglyph substitution, ROT13, and common character-level obfuscation—before the query reaches the intent classifier or the model. A lightweight anomaly detector flags inputs with unusual character distributions for human inspection. Directly motivated by the encoding bypass vulnerability documented in [6] and [28], which identify this as the highest-success-rate attack category. This layer offers the best coverage-to-latency ratio among all IAJDA layers (see Figure 7).

7.2.3. L3: Context Integrity Verifier

A classifier scanning the full context window for patterns consistent with known injection techniques: dense sequences of harmful request-response pairs consistent with many-shot priming [1], tokens matching system-prompt override syntax, and persona-replacement patterns associated with DAN-style attacks [26]. Positive detection triggers query rejection, targeted context truncation, or escalation to human review depending on confidence.

7.2.4. L4: Aligned Model Core

The language model, trained with Constitutional AI [2] and DPO [21]. Within the IAJDA, the model is one component among eight rather than the primary enforcement mechanism, removing the implicit pressure for alignment training to be impenetrable on its own—which published benchmarks confirm it currently is not.

7.2.5. L5: Output Scorer

A classifier evaluating completed model generations on harm probability and instruction-compliance score. The VIR signal adds a third lightweight feature: responses with anomalously high token counts are flagged for additional review. Motivated by the observation that model-level alignment failures produce characteristic over-hedging behavior [19].

7.2.6. L6: Tool-Use Policy Engine

A policy module governing which tool calls are permitted, under what conditions, and with what parameters. The minimal implementation uses a static allowlist and denylist; more complete versions apply runtime constraint checking and require confirmation for high-risk actions. All tool outputs pass through L1 and L2 before reaching the model context, treating external content as untrusted—directly motivated by the indirect prompt injection findings of [10].

7.2.7. L7: Trajectory Monitor

An asynchronous component tracking conversation history for signs of gradual escalation: semantic drift in harm-relevant embedding dimensions, rising harm-category classifier scores, and VIR trends. Operates asynchronously at zero synchronous latency cost. Motivated by multi-turn erosion attacks and the trajectory-level propagation of injected instructions documented in [10] and [7].

7.2.8. L8: Audit and Human Escalation

A logging and review system recording model inputs, outputs, classifier scores, and policy decisions. Cases are routed to human review with probability proportional to risk score. Human judgments feed back into updates to L1–L4 on a scheduled cadence, providing a path for handling novel attack families not covered at deployment.

7.3. Coverage vs. Latency Analysis

Figure 7 presents the tradeoff between per-layer attack coverage and added synchronous latency, with coverage estimates grounded in the benchmark failure mode analysis.

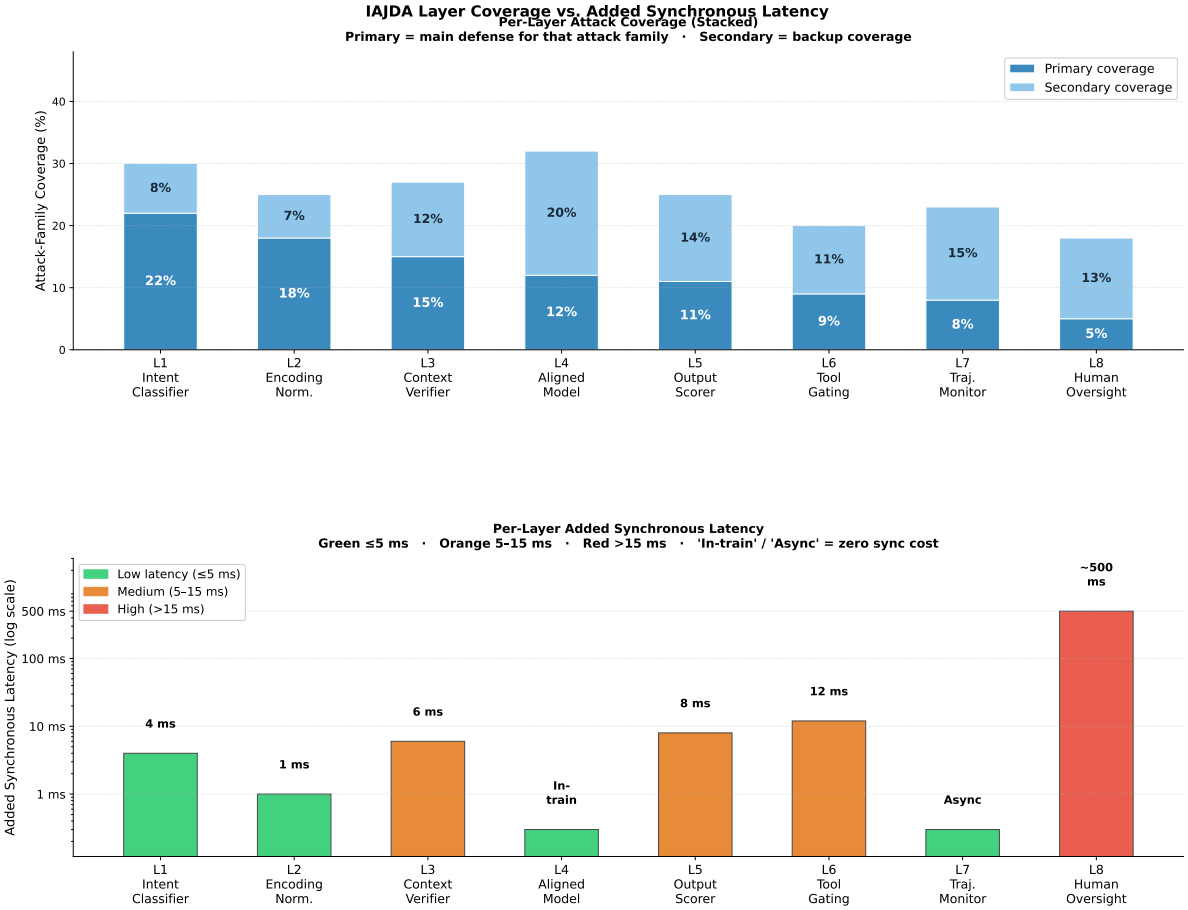

Figure 7: Top: per-layer attack-family coverage stacked as primary and secondary. Values are qualitative estimates derived from the mechanistic failure mode analysis in Section 5. Bottom: added synchronous latency per layer (log scale). Layer 2 (encoding normalization) offers the best coverage-to-latency ratio, addressing the highest-ASR attack category at ∼1 ms cost. Layers 7 and 8 operate asynchronously or with high latency, making them unsuitable as synchronous filters but valuable for retrospective analysis and adaptive improvement.

Table 7 summarizes projected improvements per attack family, framed as qualitative estimates motivated by the benchmark analysis.

Table 7: Projected improvement per attack family from IAJDA deployment (qualitative estimates grounded in published failure mode analysis)

Attack family	Primary IAJDA layer(s)	Published ASR	Projected direction
Encoding bypass	L2 (normalization)	Highest category	Major reduction
GCG adv. suffix	L5 (output scorer)	56–88%	Moderate reduction
Agent delegation	L6, L9	Context-dep.	Substantial reduction
Many-shot priming	L2, L3, L7	∼61% at 256 shots	Substantial reduction
Multi-turn erosion	L7, L9	Context-dep.	Moderate reduction
DAN / opp.-day	L2, L3	∼25–45%	Moderate reduction
Role-play framing	L1, L4	∼20–40%	Moderate reduction
Hypothetical	L1, L4	∼20–35%	Moderate reduction
Prompt injection	L2, L6	Context-dep.	Substantial reduction
Direct request	L4 (model)	∼5–17%	Minor reduction

Published ASR values from HarmBench [16], Zou et al. [31], and Anil et al. [1]. "Projected direction" is a qualitative estimate pending empirical validation of deployed IAJDA components.

8. Discussion

8.1. Scale and Safety: What Published Benchmarks Show

Larger models within each family consistently perform better on published safety benchmarks. HarmBench [16] reports substantially lower ASR for Llama-2-70B-Chat compared to Llama-2-7B-Chat on most attack categories. Gemma-2-27B outperforms Gemma-2-9B on published safety metrics, consistent with the Gemma-2 technical report. However, Gemma-2-27B also outperforms Llama-3-70B on composite safety measures despite having fewer parameters, which is consistent with the CAI/RLAIF alignment approach providing a safety advantage over RLHF+DPO at comparable scale. Since model families differ along many training dimensions simultaneously, this observation is correlation, not a controlled comparison of alignment methods.

8.2. Encoding Bypass: The Most Actionable Gap

The consistent identification of encoding bypass as the highest-success-rate attack category across survey literature [6, 28] points to a practically addressable problem. Input normalization of the kind described in IAJDA Layer 2 would convert an encoded request to its plain-text equivalent before it reaches any other component. The gap persists because classifier-based moderation systems are trained on natural-language text, while large language models learn to handle encoded formats during pretraining. This is a specific, well-defined engineering problem with a clear solution: normalization preprocessing at low latency cost.

8.3. Agentic Deployment: A Structurally Different Problem

The published results from [10] and [23] confirm that agentic safety failures differ in kind from direct-request safety failures. A model that handles direct harmful requests well—as reflected in HarmBench baseline performance—may still fail badly in agentic settings because the attack arrives through the tool layer. Infrastructure-level controls (IAJDA Layers 6, 7, 9) are not optional for agentic safety.

8.4. VIR as a Monitoring Signal: A Hypothesis

The VIR metric proposed in this paper is motivated by published observations about RLHF-trained model hedging behavior [19], but it has not yet been validated against published benchmark datasets. We offer it as a testable hypothesis: if adversarial prompts disproportionately produce hedged responses, then token count is a lightweight adversarial intent signal. Systematic validation on HarmBench or JailbreakBench prompt distributions is identified as a future work priority.

8.5. Alignment Is Necessary But Not Sufficient

Published benchmark results across HarmBench, JailbreakBench, and GCG evaluations consistently show that all current aligned models achieve non-trivial attack success rates under adversarial conditions. This is not a critique of alignment research; it is an observation that alignment training at its current stage does not cover the full threat landscape. Where alignment performs well, the inputs tend to be direct harmful requests well represented in alignment training data. Where it fails, inputs involve distribution shifts, encoding obfuscation, or deployment contexts that differ structurally from alignment training coverage. A defense-in-depth approach, where the model is one enforcement layer among several, fits this pattern better than relying on alignment alone.

9. Limitations

Benchmark coverage and recency. The published benchmarks used in this analysis (HarmBench, JailbreakBench, AdvBench) were released in 2023–2024 and may not fully characterize the safety profiles of the latest model versions. Models are continuously updated, and safety evaluations represent point-in-time snapshots.

Model family differences. The safety comparisons in this paper are between different model families trained with different data, methods, and objectives. Observed differences in safety profiles are consistent with known alignment methodology differences but should not be interpreted as controlled experiments.

English-language focus. Available evaluation benchmarks are predominantly English-language. Cross-lingual safety properties are not captured by the current benchmark synthesis.

VIR not yet benchmarked. The Verbosity Inflation Ratio is proposed in this paper as a monitoring signal but has not been validated against any published benchmark dataset. It is an untested hypothesis.

IAJDA projections are qualitative. Coverage and latency estimates for IAJDA layers are motivated by mechanistic analysis rather than empirical measurement of deployed components.

Agentic benchmarks are limited. Published systematic benchmarks for agentic safety (instruction hierarchy, indirect injection, multi-agent propagation) are less mature than direct request benchmarks. Published qualitative findings [10, 23] provide grounding but not the same precision as HarmBench-style evaluations.

10. Future Work

The most immediate priority is empirical validation of the IAJDA layer components through systematic evaluation on HarmBench, JailbreakBench, and AdvBench, with each layer implemented as a modular wrapper around open-weight model inference.

Several other directions are worth pursuing. Systematic measurement of refusal rate degradation as a function of context position would fill a specific gap in the existing benchmark literature, providing direct quantitative confirmation of the Liu et al. positional effect in the safety domain. Extending the VIR hypothesis to published benchmark datasets would validate or refute its utility as a monitoring signal. Cross-lingual benchmark extension would address the English-language limitation. Agentic safety benchmarks analogous to HarmBench in rigor and coverage are needed to support the kind of systematic evaluation currently possible only for direct-request safety. Finally, formal methods for reasoning about what safety properties hold when specific SAP layers are composed in sequence would improve the scientific rigor of layered defense design.

11. Conclusion

This paper characterized how LLM safety enforcement has evolved, synthesized published benchmark evidence for where current open-weight models stand on relevant safety dimensions, and proposed a more deliberately designed defense architecture grounded in that evidence.

Published benchmark results tell a consistent story. Models handle direct harmful requests reasonably well, consistent with HarmBench results showing 5–17% ASR on direct requests for instruction-tuned models. Attack success rises substantially under GCG adversarial suffix optimization (56–88% ASR, Zou et al.), many-shot priming (∼61% at 256 shots, Anil et al.), and encoding bypass (identified as the highest-success-rate category in survey literature). Agentic deployment introduces qualitatively different vulnerabilities demonstrated empirically by Greshake et al. and Ruan et al.

The encoding bypass gap is the most actionable near-term target: survey literature consistently identifies it as the highest-success attack category, and input normalization addresses it at low latency cost. The agentic vulnerability gap requires infrastructure-level controls rather than additional alignment training.

Gemma-2-27B demonstrates the strongest safety profile across published benchmarks. The comparison with Llama-3-70B is consistent with a CAI/RLAIF alignment advantage, though the result is confounded by other training differences.

The IAJDA is a proposed architecture motivated by published failure mode analysis. Empirical validation of each layer's contribution is the primary outstanding work.

Alignment training is necessary and must continue to advance. But published benchmarks confirm that alignment alone does not cover the full threat landscape at its current stage of development. A pipeline that distributes enforcement across training, runtime classification, policy control, and monitoring fits the observed pattern of coverage gaps better than one that expects the model to carry most of that responsibility.

Acknowledgements

The author acknowledges Dr. Sunaina Kaur, PhD, for extensive mentorship, technical discussions, editorial guidance, and research support during the development of this work.

References

[1] Anil, C., Durmus, E., Sharma, M., et al. (2024). Many-shot jailbreaking. *Advances in Neural Information Processing Systems* (NeurIPS 2024). Anthropic Technical Report.

[2] Bai, Y., Kadavath, S., Kundu, S., et al. (2022). Constitutional AI: Harmlessness from AI feedback. *arXiv:2212.08073*.

[3] Bai, Y., Jones, A., Ndousse, K., et al. (2022). Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv:2204.05862*.

[4] Chao, P., Robey, A., Dobriban, E., et al. (2024). JailbreakBench: An open robustness benchmark for jailbreaking large language models. *arXiv:2404.01318*.

[5] Christiano, P., Leike, J., Brown, T., et al. (2017). Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems* 30 (NeurIPS 2017).

[6] Chu, Z., Liu, Z., Chen, H., et al. (2024). A comprehensive study of jailbreak attacks and defenses for large language models. *arXiv:2402.13457*.

[7] Cohen, R., Barak, M., et al. (2024). Here comes the AI worm: Unleashing zero-click worms that target GenAI-powered applications. *arXiv:2403.02817*.

[8] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proc. NAACL-HLT 2019*.

[9] Gao, L., Biderman, S., Black, S., et al. (2023). Scaling laws for reward model overoptimization. *arXiv:2210.10760*.

[10] Greshake, K., Abdelnabi, S., Mishra, S., et al. (2023). Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. *Proc. 16th ACM Workshop on Artificial Intelligence and Security* (AISec@CCS 2023), pp. 79–90. doi:10.1145/3605764.3623985.

[11] Inan, H., Upasani, K., Chi, J., et al. (2023). Llama Guard: LLM-based input-output safeguard for human-AI conversations. *arXiv:2312.06674*.

[12] Kwon, W., Li, Z., Zhuang, S., et al. (2023). Efficient memory management for large language model serving with PagedAttention. *Proc. ACM SOSP 2023*.

[13] Lee, H., Phatale, S., Mansoor, H., et al. (2023). RLAIF: Scaling reinforcement learning from human feedback with AI feedback. *arXiv:2309.00267*.

[14] Lees, A., Tran, V., Tay, Y., et al. (2022). A new generation of Perspective API: Efficient multilingual character-level transformers. *Proc. KDD 2022*.

[15] Liu, N.F., Lin, K., Hewitt, J., et al. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12, 157–173. doi:10.1162/tacl_a_00638.

[16] Mazeika, M., Phan, L., Yin, X., et al. (2024). HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv:2402.04249*.

[17] Meta AI (2023). Llama Guard model card. *Meta AI Research*.

[18] OpenAI (2023). OpenAI Moderation API. *OpenAI Technical Documentation*.

[19] Ouyang, L., Wu, J., Jiang, X., et al. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (NeurIPS 2022). arXiv:2203.02155.

[20] Perez, F. and Ribeiro, I. (2022). Ignore previous prompt: Attack techniques for language models. *ML Safety Workshop, NeurIPS 2022*. arXiv:2211.09527.

[21] Rafailov, R., Sharma, A., Mitchell, E., et al. (2023). Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems* 36 (NeurIPS 2023). arXiv:2305.18290.

[22] Rebedea, T., Dinu, R., Sreedhar, M., et al. (2023). NeMo Guardrails: A toolkit for controllable and safe LLM applications with programmable rails. *Proc. EMNLP 2023 System Demonstrations*.

[23] Ruan, Y., Dong, H., Wang, A., et al. (2024). Identifying the risks of LM agents with an LM-emulated sandbox. *arXiv:2309.15817*.

[24] Schulman, J., Wolski, F., Dhariwal, P., et al. (2017). Proximal policy optimization algorithms. *arXiv:1707.06347*.

[25] Wei, J., Bosma, M., Zhao, V., et al. (2021). Finetuned language models are zero-shot learners. *arXiv:2109.01652*.

[26] Wei, A., Haghtalab, N., and Steinhardt, J. (2023). Jailbroken: How does LLM safety training fail? *Advances in Neural Information Processing Systems* 36 (NeurIPS 2023). arXiv:2307.02483.

[27] Yang, X., Wang, X., Zhang, N., et al. (2023). Shadow alignment: The ease of subverting safely-aligned language models. *arXiv:2310.02949*.

[28] Yi, J., Ye, C., Chen, J., et al. (2024). Jailbreak attacks and defenses against large language models: A survey. *arXiv:2407.04295*.

[29] Zhu, Y., Kiros, R., Zemel, R., et al. (2015). Aligning books and movies: Towards story-like visual explanations. *Proc. ICCV 2015*.

[30] Ziegler, D., Stiennon, N., Wu, J., et al. (2019). Fine-tuning language models from human preferences. *arXiv:1909.08593*.

[31] Zou, A., Wang, Z., Kolter, J.Z., and Fredrikson, M. (2023). Universal and transferable adversarial attacks on aligned language models. *arXiv:2307.15043*.

A. Benchmark Source Summary

Table 8: Primary benchmark sources used in this analysis

Benchmark	Citation	Attack types	Key finding used
HarmBench	[16]	Direct, GCG, PAIR, AutoDAN	Direct ASR 5–17% for inst.-tuned models
JailbreakBench	[4]	PAIR, GCG, 100 behaviors	Framing attacks substantially higher ASR than direct
AdvBench	[31]	GCG adversarial suffix	56–88% ASR on aligned open-weight models
Many-shot study	[1]	Many-shot priming	∼61% ASR at 256 shots, power-law scaling
TACL long-context	[15]	Position effects	U-shaped performance; middle positions under-weighted
Indirect injection	[10]	Tool-output injection	Demonstrated on GPT-4+Bing; bypasses input defenses
Jailbreak survey	[28]	All categories	Encoding bypass ranked as highest-ASR category
Comprehensive survey	[6]	All categories	Encoding bypass consistently highest success rate

B. Companion Repository Structure

```
sap-llm-safety/
  paper/
    main.tex                    LaTeX source
    figures/                    All figures (PDF + PNG)
      gen_figures.py            Figure generation script
  framework/
    sap_layers.yaml             Layer definitions
    iajda_spec.yaml             IAJDA component specifications
  benchmark_synthesis/
    harmbench_analysis.py       HarmBench result extraction
    jailbreakbench_analysis.py  JailbreakBench result extraction
    gcg_results.py              GCG/AdvBench result synthesis
    manyshot_scaling.py         Many-shot power-law analysis
  eval/
    exp_vir_validation/         VIR hypothesis validation scripts
    exp_context_position/       Context-position safety measurement
    exp_iajda_components/       Per-layer IAJDA validation
  docs/
    setup.md                    Environment setup
    reproduce.md                Reproduction guide
  requirements.txt
  README.md
```