GUARDALIGN: GUARDRAIL-AWARE SAFETY ALIGNMENT FOR MODERN LLM PIPELINES

Anonymous authors
Paper under double-blind review

ABSTRACT

External guardrails increasingly complement aligned LLMs against jailbreaks, yet guardrails and backend models are typically optimized independently. Consequently, an unsafe prompt missed by an input guardrail reaches a backend never explicitly aligned to the guardrail's coverage gaps. We introduce Guardrail-Aware Safety Alignment (GuardAlign), a paradigm that retains the full safety preference dataset and augments each example with offline decisions and normalized unsafe-confidence evidence from multiple input guardrails. Within GuardAlign, Guardrail-Aware Preference Optimization (GuardPO) is a reference-free objective that replaces a static margin with an example-specific target margin. Its Detection Penalty emphasizes guardrail misses and low-confidence detections, while its Stubbornness Penalty targets current-policy misrankings only when safety and helpfulness agree. Across four LLM backbones and six safety test sets, every GuardAlign variant attains a lower overall mean pipeline attack success rate than Pre-Guardrail, Post-Guardrail, and Self-Refine; GuardPO performs best at 2.42%, versus 7.23% for the strongest of these baselines. Without an inference-time guardrail, GuardPO also reduces adaptive-attack ASR to 23.75%, compared with 31.00% for G-SimPO and 43.75% for G-DPO. It further incurs substantially less online overhead than output filtering and iterative refinement, while retaining the most BIG-Bench Hard utility among the evaluated guardrail-aware objectives.

1 INTRODUCTION

Despite substantial progress in safety alignment (Bai et al., 2022), large language models (LLMs) remain vulnerable to jailbreak attacks that elicit policy-violating responses (Wei et al., 2023; Shen et al., 2024; Paulus et al.; Chao et al., 2024; Zou et al., 2023; Liu et al., 2024; 2025a; Li et al., 2024). Deployments therefore increasingly place an external guardrail around an instruction-tuned backend LLM. In a pre-guardrail pipeline, the guardrail filters user prompts before generation (Inan et al., 2023; Lin et al., 2023; Liu et al., 2025b); in a post-guardrail pipeline, it screens generated responses before release (Zhao et al., 2025; Han et al., 2024; Zeng et al., 2024). This modular design makes guardrails easy to deploy or replace, but pipeline safety depends on how the two components interact rather than on either component alone.

We study an overlooked consequence of this modularity: guardrails and backend LLMs are typically optimized independently. For a verified unsafe prompt in an input-filtering pipeline, a harmful output requires two consecutive failures—the guardrail must pass the prompt, and the backend must then answer unsafely. Conventional safety alignment, including supervised fine-tuning, reinforcement learning from human feedback, and preference optimization (He et al.; Zheng et al.; Song et al., 2025; Christiano et al., 2017; Rafailov et al., 2023; Meng et al., 2024; Wu et al.), strengthens standalone refusal behavior but does not condition training on guardrail coverage. It therefore does not explicitly distinguish prompts that are confidently detected, detected with weak evidence, or missed altogether, even though these cases expose different residual risks after deployment. Training only on guardrail misses would also discard well-covered examples that remain useful safety anchors and ignore graded confidence evidence. The resulting *compatibility gap* leaves the backend policy poorly adapted to the upstream guardrail's coverage gaps precisely when it becomes the final opportunity to prevent harm.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

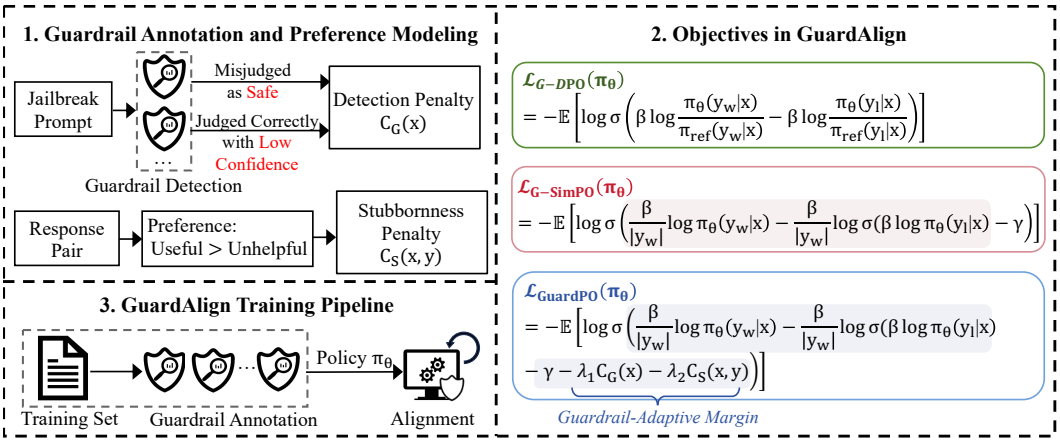

Figure 1: Overview of GuardAlign. ① Offline guardrail decisions and normalized unsafe-confidence scores determine the prompt-level Detection Penalty $C_G(x)$, while helpfulness agreement gates the response-level Stubbornness Penalty $C_S(x, y)$ derived from the current policy ordering. ② G-DPO and G-SimPO apply standard preference objectives to the augmented data, whereas GuardPO incorporates both penalties into an example-specific margin. ③ The augmented preference data align the backend policy π_θ. These annotations are used only during training, while the deployed input guardrail continues to filter prompts at inference time. Here, x is a verified unsafe prompt and $y = (y_w, y_l)$ comprises a safe chosen response y_w and an unsafe rejected response y_l.

Deployment-side remedies address this residual risk only at inference time. Post-generation filtering must generate and inspect a response before release, while iterative refinement invokes the backend repeatedly to critique and revise its output. These mechanisms provide additional opportunities to intercept harm, but add online computation and remain limited by imperfect safety judgments. We instead ask whether guardrail behavior can serve as a training signal, allowing the backend to compensate for upstream coverage gaps without adding runtime stages.

To close this gap, we propose *Guardrail-Aware Safety Alignment* (GuardAlign), a training paradigm that turns guardrail behavior into offline supervision for backend alignment. Rather than changing the inference architecture or discarding detected prompts, GuardAlign retains the full safety preference dataset and augments each unsafe prompt with native decisions and normalized unsafe-confidence evidence from multiple input guardrails. Because raw logit scales differ across guardrails, we standardize each guardrail's safe–unsafe logit margin over the training data and map it to a bounded evidence score, while retaining the native decision to identify misses. Our annotating guardrails are Llama-Guard-4-12B, ShieldGemma-9B, Qwen3Guard-8B-Gen, and WildGuard-7B. Missed and uncertain prompts identify where stronger correction is needed, while confidently detected prompts remain safety anchors under the base preference objective. The annotations are used only during training; the deployed guardrail–LLM architecture remains unchanged.

Crucially, GuardAlign treats these annotations as risk evidence rather than replacement preference labels. A correct decision by one guardrail cannot erase a miss by another, because such a miss exposes a coverage gap that could reach the backend. Conversely, agreement among guardrails does not always imply equal certainty: high-confidence agreement provides a stable safety anchor, whereas low-confidence agreement still signals residual risk. Retaining all three cases allows GuardAlign to allocate additional optimization pressure without replacing base preference supervision or filtering the dataset. Because annotation occurs offline, complementary guardrails can expose different blind spots without adding model calls to deployment. The remaining challenge is to translate this evidence into a preference objective that also accounts for evolving policy behavior and safety–helpfulness conflicts.

Within GuardAlign, we introduce *Guardrail-Aware Preference Optimization* (GuardPO), a reference-free objective that replaces a static target margin with an example-specific, guardrail-aware gap. Its prompt-level *Detection Penalty* increases the target margin for prompts missed by any guardrail and, more conservatively, for prompts detected by all guardrails with low aggregate

unsafe confidence. Its response-level *Stubbornness Penalty* activates when the current policy ranks the unsafe response above the safe response, but only when independent safety and helpfulness preferences agree on the safe response. The policy-dependent term is detached within each step, so it adjusts the target margin without creating an additional gradient path. By separating prompt-level coverage risk from response-level policy misranking, GuardPO concentrates training pressure on consequential examples without requiring a reference policy or an auxiliary reward or cost model.

The two penalties play complementary roles during optimization. The Detection Penalty is fixed and represents environment-side exposure independent of the current backend, whereas the Stubbornness Penalty is recomputed as the policy evolves. Adapting the margin rather than filtering or relabeling examples preserves every pair's original preference direction and allows the base objective to train well-covered samples. When a prompt both exposes a guardrail coverage gap and remains misranked by the policy, the penalties combine; when the policy ordering is correct or helpfulness conflicts with safety, the dynamic penalty vanishes. This construction avoids treating guardrail decisions as ground-truth preference labels while preventing the policy-dependent quantity from creating an unintended optimization shortcut.

We evaluate GuardAlign on six safety test sets and an adaptive jailbreak attack using four open-weight backbones: Llama-3.1-8B-Instruct, Qwen3-8B, Mistral-7B-Instruct-v0.3, and Gemma-2-9B-IT (Meta, 2026; Team, 2026; 2024; Google, 2026). Our evaluation separates end-to-end pipeline protection from standalone backend robustness: the former retains the inference-time guardrail, whereas the latter attacks the aligned LLM directly. This distinction tests whether safety gains persist after removing the filter rather than merely reflecting more aggressive blocking. Every GuardAlign variant attains a lower six-set mean attack success rate (ASR) than Pre-Guardrail on each backbone. GuardPO achieves 2.42% average pipeline ASR, compared with 7.23% for Pre-Guardrail, 9.56% for Post-Guardrail, and 12.63% for Self-Refine. Under direct adaptive attacks without an inference-time guardrail, GuardPO reaches 23.75% ASR, outperforming G-SimPO (31.00%) and G-DPO (43.75%). Post-Guardrail and Self-Refine incur 6.89× and 32.73× GuardPO's online overhead, respectively. Ablations validate both penalties and the all-correct confidence branch, while BIG-Bench Hard results show that GuardPO retains the most utility among the evaluated guardrail-aware objectives, with a backbone-dependent alignment tax.

Our contributions are threefold:

- We identify the compatibility gap between independently optimized guardrails and backend LLMs, and propose GuardAlign to integrate offline guardrail decisions and normalized unsafe-confidence evidence into backend safety alignment (§3).
- We develop GuardPO, a reference-free objective whose Detection and Stubbornness Penalties adapt the training margin to guardrail coverage risk and current policy misranking (§3).
- Across four LLM families, extensive experiments show that GuardAlign improves end-to-end pipeline safety and that GuardPO provides the strongest adaptive robustness and utility retention among the evaluated guardrail-aware objectives (§4).

2 PRELIMINARIES

2.1 THREAT MODEL: GUARDRAIL–LLM PIPELINES

Pipeline and Failures. We consider a two-stage deployment in which an input guardrail first classifies a user prompt and forwards only prompts labeled *Safe* to a backend policy π_θ. This input-filtering configuration provides prompt-level guardrail evidence for aligning the backend policy. To place this design in the broader landscape of guardrail-augmented deployment, we also evaluate post-generation filtering as an alternative pipeline paradigm in §4. For a verified unsafe prompt, a pipeline-level safety failure requires both stages to fail: the guardrail must incorrectly pass the prompt, and the backend policy must subsequently generate a policy-violating response (Wei et al., 2023; Wang et al., 2026; Dong et al., 2025). Our focus is thus the residual risk created by guardrail false negatives, for which the backend policy is the final opportunity to prevent a harmful output.

Attacker Capabilities. We assume a black-box adaptive attacker who controls the prompt, knows the pipeline architecture, and can observe whether each request is blocked or answered. The attacker

has no access to the deployed models' parameters, internal activations, or guardrail-side confidence signals, but may use a public model or a local surrogate to construct transferable attacks.

Defender and Training Data. The defender controls and fine-tunes the backend policy π_θ. Let $\mathcal{G} = \{G_j\}_{j=1}^{M}$ denote a set of input guardrails whose decisions and logits are collected offline to annotate a safety preference dataset $\mathcal{D}$. Each example $(x, y_w, y_l) \in \mathcal{D}$ consists of a verified unsafe prompt x, a safe chosen response y_w, and an unsafe rejected response y_l, such that $y_w \succ_{\mathrm{s}} y_l$ under the safety preference $\succ_{\mathrm{s}}$. The responses are also annotated under an independent helpfulness preference $\succ_{\mathrm{h}}$. We write $h = \mathbb{I}[y_w \succ_{\mathrm{h}} y_l]$ to indicate whether safety and helpfulness agree on the chosen response. The guardrail annotations and the helpfulness indicator augment the preference examples only during training; they are not provided to π_θ at inference time, and the deployed guardrail–LLM architecture remains unchanged.

2.2 Reference-Free Preference Optimization

Preference Objectives. Given $\mathcal{D}$, DPO (Rafailov et al., 2023) learns from pairwise preferences through the implicit reward $r_\theta(x, y) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\mathrm{ref}}(y|x)}$, which measures the policy against a fixed reference policy π_{ref}. SimPO (Meng et al., 2024) removes this reference model and scores each response directly using its length-normalized log-likelihood, $\ell_\theta(x, y) = |y|^{-1} \log \pi_\theta(y \mid x)$. Its objective is

$$\mathcal{L}_{\mathrm{SimPO}}(\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma(\beta \left[\ell_\theta(x, y_w) - \ell_\theta(x, y_l)\right] - \gamma)\right]. \tag{1}$$

Here, $\beta > 0$ controls the scale of the preference difference, while γ is a target margin that encourages the policy to score y_w above y_l by a prescribed amount.

From a Static to a Risk-Adaptive Margin. In Equation 1, the same target margin γ is applied to every preference pair, irrespective of whether the unsafe prompt was confidently detected, detected with low confidence, or missed by the guardrails. It also does not distinguish examples in which safety and helpfulness agree from those in which the two preferences conflict. Consequently, the objective cannot allocate stronger updates to examples that expose greater residual risk in the deployed pipeline. The method introduced in the next section (§3) replaces γ with an example-specific target derived from the offline guardrail evidence, the safety–helpfulness relation, and the current policy ordering. For the policy-dependent component, we apply stop-gradient so that it adjusts the target margin without introducing an additional gradient path.

3 Guardrail-Aware Preference Optimization

As discussed in §2.2, a uniform preference margin treats all training pairs alike even though their residual deployment risks differ substantially. We therefore introduce the *Guardrail-Aware Safety Alignment* (GuardAlign) paradiam, illustrated in Figure 1. Rather than training only on prompts that bypass a guardrail, GuardAlign retains the full safety preference dataset and augments each example with offline decisions and confidence evidence from multiple guardrails. The guardrail outputs serve as auxiliary risk evidence rather than as preference labels: confidently detected prompts remain safety anchors, whereas missed and uncertain prompts identify where the backend policy requires stronger correction. Within this paradigm, *Guardrail-Aware Preference Optimization* (GuardPO) converts the augmented evidence into an example-specific target margin. It combines a prompt-level *Detection Penalty*, which captures guardrail misses and uncertainty, with a response-level *Stubbornness Penalty*, which captures incorrect policy rankings when safety and helpfulness agree.

3.1 Guardrail-Aware Implicit-Reference Gap

The SimPO objective in Equation 1 encourages the scaled score difference $\beta[\ell_\theta(x, y_w) - \ell_\theta(x, y_l)]$ to exceed the fixed margin γ. This margin therefore acts as an implicit comparison baseline between the safe chosen response y_w and the unsafe rejected response y_l. To make this baseline risk-sensitive, GuardPO replaces γ with a *guardrail-aware implicit-reference gap*. Let $y = (y_w, y_l)$ denote the response pair. Building on the adaptive-margin view of AlphaDPO (Wu et al.), we define

$$\Delta_G(x, y) \equiv \gamma_{\mathrm{base}} + \lambda_1 C_G(x) + \lambda_2 C_S(x, y), \tag{2}$$

where γ_{base} is the base reference-free margin, $C_G(x)$ measures prompt-level risk from guardrail evidence, and $C_S(x, y)$ measures response-level misalignment under the current policy. The coefficients $\lambda_1, \lambda_2 \geq 0$ control their respective contributions. A larger $\Delta_G(x, y)$ imposes a stricter separation target on examples that require greater correction. Because this risk-dependent gap introduces no reference policy, GuardPO remains a reference-free objective.

3.2 GUARDRAIL-ADAPTIVE MARGIN

The adaptive gap combines two complementary quantities. The Detection Penalty is computed from offline guardrail annotations and remains fixed during policy training, whereas the Stubbornness Penalty is recomputed from the current policy at each training step.

Detection Penalty. For the guardrail set defined in §2.1, let $d_j \in \{\text{Safe}, \text{Unsafe}\}$ be the native decision of G_j, and let $a_j^s(x)$ and $a_j^u(x)$ be its safe and unsafe logits for prompt x. Because logit scales differ across guardrails, we first define the unsafe logit margin $m_j(x) = a_j^u(x) - a_j^s(x)$ and standardize it over all prompts in $\mathcal{D}$. Let μ_j and $\sigma_j > 0$ be the population mean and standard deviation of $m_j(x)$ on $\mathcal{D}$. We then compute the normalized unsafe-confidence score

$$p_j(x) = \text{sigmoid}\left(\frac{m_j(x) - \mu_j}{\sigma_j}\right). \tag{3}$$

This transformation places heterogeneous logits on a common bounded scale without requiring benign calibration data; $p_j(x)$ is a relative evidence score, not a calibrated probability. Also, the native decision d_j, rather than a new threshold on $p_j(x)$, determines whether G_j misses the unsafe prompt.

Accordingly, $\mathbb{I}[d_j = \text{Safe}]$ identifies a false negative, and $1 - p_j(x)$ increases as that guardrail's normalized unsafe evidence decreases. We assign each guardrail a weight $w_j > 0$, with $\sum_j w_j = 1$, and define

$$C_G(x) = \begin{cases} \displaystyle\sum_j w_j \mathbb{I}[d_j = \text{Safe}](1 - p_j(x)), & \exists j : d_j = \text{Safe}, \\ \alpha \max\left(0, \dfrac{\tau_G - \sum_j w_j p_j(x)}{\tau_G}\right), & \forall j : d_j = \text{Unsafe}. \end{cases} \tag{4}$$

The first branch accumulates evidence only from guardrails that miss the prompt, so a correct decision cannot cancel a false negative. The second branch applies when every guardrail is correct and measures the shortfall of their weighted unsafe confidence from a target $0 < \tau_G \leq 1$. The factor $0 \leq \alpha < 1$ attenuates this all-correct confidence penalty relative to the miss penalty. Hence, high-confidence detections receive no additional penalty, but they remain in the dataset and are still optimized under the base margin. Because $p_j(x) \in (0, 1)$ and the weights sum to one, $0 \leq C_G(x) \leq 1$.

Stubbornness Penalty. When $h = 1$, the safe response y_w is preferred under both safety and helpfulness, so the desired policy ordering is $\ell_\theta(x, y_w) \geq \ell_\theta(x, y_l)$. We measure the one-sided violation of this boundary as

$$C_S(x, y) = \begin{cases} \text{sg}[\max(0, \ell_\theta(x, y_l) - \ell_\theta(x, y_w))], & h = 1, \\ 0, & h = 0, \end{cases} \tag{5}$$

where $\text{sg}[\cdot]$ denotes stop-gradient: it preserves the forward value but has zero derivative. Thus, $C_S(x, y) > 0$ only when safety and helpfulness agree on y_w but the policy ranks y_l higher; the penalty grows with the magnitude of this misranking. When $h = 0$, we impose no Stubbornness Penalty because helpfulness does not also favor y_w; the example still contributes through the base preference objective. We recompute C_S from the policy at each training step, but detach it within that step so that it changes only the target margin and creates no additional gradient path.

3.3 THE GUARDPO OBJECTIVE

Substituting the adaptive gap in Equation 2 into the reference-free objective yields

$$\begin{aligned} \mathcal{L}_{GuardPO}(\theta) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \Big[\log \sigma \Big(\beta \left[\ell_\theta(x, y_w) - \ell_\theta(x, y_l) \right] \\ - \gamma_{\text{base}} - \underbrace{\lambda_1 C_G(x)}_{\text{Detection Penalty}} - \underbrace{\lambda_2 C_S(x, y)}_{\text{Stubbornness Penalty}} \Big) \Big]. \end{aligned} \tag{6}$$

For a fixed score difference, either penalty lowers the sigmoid argument, strengthening the update toward a higher relative likelihood of y_w. Their effects are additive when a prompt exposes both guardrail-level risk and policy-level misranking. Setting $\lambda_1 = \lambda_2 = 0$ recovers the SimPO objective with margin γ_{base}. Together, GuardAlign augments the available training evidence, while GuardPO converts that evidence and the current response ordering into a risk-adaptive optimization target. The resulting framework retains well-covered samples as safety anchors, emphasizes missed or uncertain prompts, avoids extra pressure from the Stubbornness Penalty when safety and helpfulness do not agree, and requires neither a reference policy nor an auxiliary reward or cost model.

4 EVALUATION

We evaluate GuardAlign as a guardrail-aware safety alignment paradigm and GuardPO as its specialized optimization objective across four open-weight LLM families. Specifically, **RQ1** in §4.2 compares GuardAlign with existing inference-time safety paradigms in end-to-end safety and efficiency; **RQ2** in §4.3 tests whether GuardPO outperforms G-SimPO and G-DPO against adaptive jailbreaks without an inference-time guardrail; **RQ3** in §4.4 isolates the safety contribution of each component; and **RQ4** in §4.5 examines how much general utility GuardAlign-aligned models retain.

4.1 EXPERIMENTAL SETUP

Datasets. We train GuardPO, G-SimPO, and G-DPO on the official PKU-SafeRLHF-30K training set (Ji et al., 2025), augmented with the guardrail annotations described in §3.2. We obtain these annotations from *Llama-Guard-4-12B*, *ShieldGemma-9B*, *Qwen3Guard-8B-Gen*, and *WildGuard-7B*; for Qwen3Guard, we map both *unsafe* and *controversial* to *Unsafe*. Safety evaluation covers six test sets: *IJP*, *GCG*, *AutoDAN*, *AutoDAN-Turbo*, *DrAttack*, and the official PKU-SafeRLHF test set. Table 3 reports the sources, sizes, and attack configurations of the first five sets. For utility, we evaluate all 6,511 available examples from the 27 *BIG-Bench Hard* (BBH) subtasks (Suzgun et al., 2023), grouped into Logical, Algorithmic, Language, and Commonsense categories.

Models and Training. We evaluate *Llama-3.1-8B-Instruct*, *Qwen3-8B*, *Mistral-7B-Instruct-v0.3*, and *Gemma-2-9B-IT*; Qwen3 runs in non-thinking mode. We fine-tune backbones for five epochs with LoRA (rank 64, scaling 64, dropout 0.05), bf16, AdamW, a cosine schedule with 10% warmup, and a 10^{-5} learning rate. Training uses four A800 GPUs, per-device batch size 2, 16 gradient-accumulation steps, gradient checkpointing, FlashAttention-2, and seed 42. The maximum sequence length is 2,048, with prompts truncated to 1,800 tokens when necessary. For GuardPO, we use uniform annotation-guardrail weights, $w_j = 1/M = 1/4$, so that no guardrail is privileged a priori. The hyperparameter search detailed in §B selects $(\lambda_1, \lambda_2, \alpha, \tau_G) = (1.00, 2.00, 0.40, 0.50)$, which we use for every backbone. G-SimPO and G-DPO use the same data, LoRA configuration, optimizer, batching, and training budget; only the preference objectives and their associated settings differ.

Baselines. We compare three GuardAlign variants with three existing deployment paradigms in §4.2. The GuardAlign variants and *Pre-Guardrail* use ShieldGemma-9B to filter prompts before generation; the former pair it with aligned backends, whereas the latter uses the original backend LLM. *Post-Guardrail* applies ShieldGemma-9B to responses from the original backend LLM. *Self-Refine* (Kim et al., 2024) follows the safety-feedback and revision procedure. For each backbone, all methods are evaluated on the same prompt sets under matched generation settings where applicable.

Adaptive Attack. We use LLM-Fuzzer (Yu et al., 2024) to evaluate adaptive-jailbreak robustness. LLM-Fuzzer iteratively mutates candidate prompts and uses target-model responses to guide subsequent mutations. We use GPT-5.5 as the mutation model and allow 200 queries per target LLM.

Metrics. We report attack success rate (ASR; ↓) and per-sample inference latency (ms/sample; ↓). An attack succeeds against an input-filtering pipeline only if the prompt passes the guardrail and the output is harmful; against a post-guardrail pipeline, the harmful output must also pass the output filter. GPT-5.5 judges whether each response is harmful. For BBH, we compute exact-match accuracy for each subtask and macro-average task-level accuracies within each capability category.

Table 1: End-to-end safety and latency across five jailbreak attacks and one adversarial test set. We report ASR (↓, %) by guardrail–LLM paradigm and additional inference latency (ms/sample; ↓). Per dataset–backbone combination, bold and underlining denote the best and second-best ASRs.

	Method	Backbone	Jailbreak Scenario (ASR)						Latency (ms/sample)
			IJP	GCG	AutoDAN	DrAttack	ADT[a]	SafeRLHF[b]	
Existing Paradigms	Pre-Guardrail	Llama	3.00	1.00	13.00	2.50	22.00	3.50	54.03
		Qwen	3.00	**0.00**	10.50	2.00	19.50	2.50	64.18
		Mistral	4.00	0.50	12.50	5.50	21.00	3.00	54.23
		Gemma	3.50	1.00	16.00	1.50	20.00	2.50	76.22
	Post-Guardrail	Llama	6.00	2.00	16.50	6.50	21.50	7.00	585.80
		Qwen	4.00	0.50	15.00	4.00	28.50	**1.50**	670.73
		Mistral	6.50	6.50	15.00	3.50	21.00	7.00	630.27
		Gemma	4.00	0.50	17.50	3.00	30.50	1.50	1,006.46
	Self-Refine	Llama	12.00	3.00	22.00	5.00	32.00	5.00	2,723.82
		Qwen	13.00	3.00	17.50	7.00	35.50	4.00	2,052.31
		Mistral	18.50	7.00	30.50	9.50	34.50	7.50	4,016.80
		Gemma	5.00	1.00	14.00	2.50	13.00	1.00	4,959.68
Guardrail-Aware Alignment Paradigm	G-DPO	Llama	6.50	0.50	14.50	2.50	16.00	1.00	65.64
		Qwen	5.00	1.50	7.00	4.00	9.50	4.50	108.51
		Mistral	7.50	0.50	8.50	4.50	12.00	1.50	98.15
		Gemma	4.00	1.00	17.50	2.00	17.00	2.00	157.09
	G-SimPO	Llama	3.00	1.00	14.50	1.00	17.00	0.00	61.02
		Qwen	3.50	1.00	3.50	4.00	**4.50**	3.50	107.23
		Mistral	4.00	1.00	7.00	6.50	7.50	0.00	93.97
		Gemma	**2.00**	0.00	15.00	0.50	18.00	1.00	154.28
	GuardPO	Llama	**2.50**	**0.50**	**3.50**	**0.00**	**11.50**	**0.00**	64.11
		Qwen	**1.50**	1.00	**2.00**	**0.00**	5.50	2.00	102.35
		Mistral	**2.00**	**0.00**	**3.50**	**1.00**	**5.00**	**0.00**	94.56
		Gemma	2.50	**0.00**	**4.00**	**0.50**	**9.00**	**0.50**	159.15

[a] ADT denotes AutoDAN-Turbo; [b] SafeRLHF denotes the PKU-SafeRLHF test set.

4.2 RQ1: GUARDALIGN VS. EXISTING PARADIGMS

Table 1 compares the three GuardAlign variants with Pre-Guardrail, Post-Guardrail, and Self-Refine across six safety test sets and four backbones. GuardPO uses our adaptive objective, whereas G-DPO and G-SimPO use standard preference objectives under the same training paradigm. Unless stated otherwise, we average over all 24 dataset–backbone combinations.

Safety Gains from Guardrail-Aware Alignment. All three GuardAlign variants improve aggregate pipeline safety over the existing paradigms. G-DPO, G-SimPO, and GuardPO achieve average ASRs of 6.27%, 4.96%, and 2.42%, respectively, compared with 7.23% for Pre-Guardrail, 9.56% for Post-Guardrail, and 12.63% for Self-Refine. Moreover, every GuardAlign variant has a lower six-set average than Pre-Guardrail on each backbone. The common benefit is most visible on AutoDAN-Turbo: averaged across backbones, the three GuardAlign variants reach 13.63%, 11.75%, and 7.75% ASR, versus 20.63%, 25.38%, and 28.75% for the three existing paradigms. These results show that guardrail-aware backend alignment provides gains across preference objectives and model families, although their magnitude depends on the objective.

Further Safety Improvements with GuardPO. Within GuardAlign, GuardPO reduces average ASR by 3.85 percentage points relative to G-DPO and by 2.54 points relative to G-SimPO. It obtains the best six-set average on every backbone and the lowest or tied-lowest ASR in 20 of the 24 individual settings. Compared with Pre-Guardrail, GuardPO lowers ASR in 23 settings and reduces the overall average from 7.23% to 2.42%. Its largest average gain over the other GuardAlign objectives occurs on AutoDAN, where GuardPO reaches 3.25% ASR, versus 11.88% for G-DPO and 10.00% for G-SimPO. Thus, the adaptive objective provides a substantial improvement beyond the shared guardrail-aware training paradigm.

Safety–Efficiency Trade-off. GuardPO incurs 105.04 ms/sample of additional latency, comparable to G-SimPO (104.13 ms) and G-DPO (107.35 ms). Post-Guardrail and Self-Refine require 723.32 and 3,438.15 ms/sample, respectively—6.89× and 32.73× the overhead of GuardPO—while producing higher ASR. Pre-Guardrail is fastest at 62.17 ms/sample; GuardPO adds 42.87 ms/sample while reducing ASR by 4.81 percentage points. Thus, GuardPO combines the strongest aggregate safety with substantially lower online overhead than output filtering or iterative refinement.

4.3 RQ2: GuardPO vs. G-SimPO vs. G-DPO

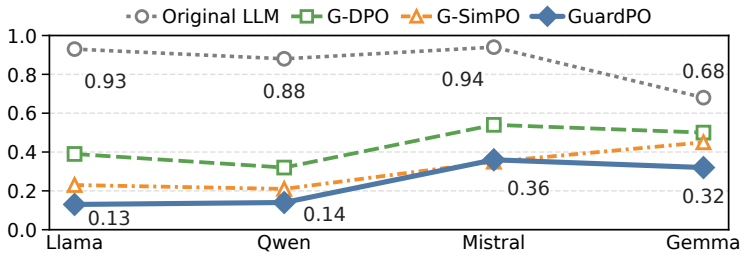

Figure 2: ASR (↓, %) for GuardAlign variants and original LLMs against adaptive LLM-Fuzzer.

To isolate the optimization objective from the shared GuardAlign training paradigm, we compare GuardPO, G-SimPO, and G-DPO under the adaptive LLM-Fuzzer attack in Figure 2. All three methods use the same guardrail-augmented preference data and matched training configurations; only their objectives differ. We query each aligned LLM directly, without an inference-time guardrail, so the model itself must reject the adaptively mutated prompts.

Overall Objective Comparison. Across the four backbones, GuardPO achieves 23.75% average ASR, compared with 31.00% for G-SimPO and 43.75% for G-DPO—reductions of 7.25 and 20.00 percentage points, respectively. All three aligned variants improve on the original models' 85.75% average ASR, while GuardPO provides the strongest adaptive robustness.

Consistency across Model Families. GuardPO outperforms G-DPO across all backbones and G-SimPO on three. The largest gap occurs on Gemma, where GuardPO obtains 32.00% ASR, versus 45.00% for G-SimPO, 50.00% for G-DPO, and 68.00% for the original model. On Llama and Qwen, GuardPO reaches 13.00% and 14.00% ASR, versus 23.00% and 21.00% for G-SimPO and 39.00% and 32.00% for G-DPO. Relative to fixed-reference G-DPO and static-margin G-SimPO, GuardPO emphasizes guardrail misses, insufficient unsafe confidence, and policy misrankings through its sample-dependent margin. Its lower ASR under response-guided mutations suggests that its training signal strengthens internal safety boundaries beyond the annotated training pairs.

4.4 RQ3: GuardPO Ablation Study

We conduct all ablations on *Gemma-2-9B-IT*, removing one component at a time while holding all other training settings fixed. Setting $\lambda_1 = 0$ or $\lambda_2 = 0$ removes the Detection or Stubbornness Penalty, respectively. Setting $\alpha = 0$ disables only the all-correct confidence branch of $C_G(x)$ while retaining its miss-based branch; τ_G is inactive in this variant.

Table 2: ASR (↓, %) of full GuardPO and component ablations. Non-ablated settings remain fixed.

Variant	Change	IJP	GCG	AutoDAN	DrAttack	AutoDAN-Turbo	PKU-SafeRLHF	Avg.
Full GuardPO	–	1.92	0.00	3.85	0.00	5.77	0.00	1.92
w/o Detection Penalty	$\lambda_1 = 0$	5.77	0.00	7.69	0.00	9.62	1.92	4.17
w/o All-correct Branch	$\alpha = 0$	3.85	0.00	11.54	0.00	11.54	1.92	4.81
w/o Stubbornness Penalty	$\lambda_2 = 0$	1.92	0.00	5.77	0.00	7.69	0.00	2.56

Complete Objective. Full GuardPO achieves the lowest average ASR at 1.92% and the lowest or tied-lowest ASR on all six test sets. Every ablation increases the average ASR, showing that both penalties and the all-correct branch contribute to the complete objective.

Component Contributions. Removing the Detection Penalty raises average ASR from 1.92% to 4.17%, including increases on IJP (1.92% to 5.77%), AutoDAN (3.85% to 7.69%), and AutoDAN-Turbo (5.77% to 9.62%). Disabling only the all-correct branch raises average ASR to 4.81% and AutoDAN and AutoDAN-Turbo ASR to 11.54%, indicating that confidence evidence from correctly detected prompts complements the miss-based branch. Removing the Stubbornness Penalty produces a smaller degradation in the average, from 1.92% to 2.56%, with increases on AutoDAN and AutoDAN-Turbo. Overall, guardrail-derived detection evidence is especially important, while policy-dependent misranking supplies an additional benefit.

4.5 RQ4: Utility Evaluation

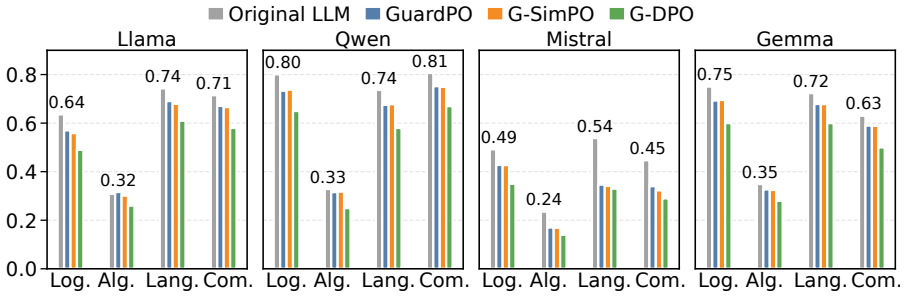

Figure 3: BBH utility of original and GuardAlign-aligned models: category-level macro-averaged accuracy. Log., Alg., Lang., and Com. denote Logical, Algorithmic, Language, and Commonsense.

Stronger safety alignment can reduce general reasoning utility, often termed the "alignment tax." We quantify this trade-off on BBH by comparing each aligned model with its original backbone.

General Utility Trade-offs. Figure 3 shows that utility changes depend on backbone and capability category. On Llama, GuardPO improves Algorithmic accuracy from 0.309 to 0.316, but reduces Logical, Language, and Commonsense accuracy from 0.636, 0.743, and 0.715 to 0.570, 0.691, and 0.671. Reductions are larger on Mistral, particularly for Language and Commonsense. GuardAlign exhibits a backbone-dependent safety–utility trade-off rather than a uniform effect on capabilities.

Comparative Utility Retention Across Objectives. The optimization objective materially affects utility retention. Across the 16 backbone–category comparisons, GuardPO outperforms G-DPO in all 16 and G-SimPO in 12, although its margins over G-SimPO are generally small. For Llama, GuardPO obtains 0.570 Logical and 0.691 Language accuracy, versus 0.559 and 0.680 for G-SimPO and 0.490 and 0.610 for G-DPO. On Mistral Commonsense, the corresponding accuracies are 0.340, 0.323, and 0.290. Overall, GuardPO retains the most utility among the evaluated guardrail-aware objectives, while still incurring a backbone-dependent alignment tax relative to the original models.

5 Conclusion

We introduced GuardAlign, a guardrail-aware safety alignment paradigm for guardrail–LLM pipelines, and its reference-free objective, GuardPO. GuardAlign retained detected and missed prompts while augmenting safety preferences with offline decisions and normalized unsafe-confidence evidence from multiple input guardrails. GuardPO made the margin example-specific through Detection and helpfulness-gated Stubbornness Penalties. Across six safety test sets, an adaptive attack, and four LLM families, GuardAlign improved pipeline safety over evaluated deployment paradigms, while GuardPO achieved the strongest average safety among guardrail-aware objectives in end-to-end and adaptive evaluations. It incurred substantially less online overhead than output filtering and iterative refinement. Ablations confirmed both penalties' contributions, while BBH revealed backbone-dependent safety–utility trade-offs and the strongest utility retention for GuardPO. Results showed that guardrail risk evidence enabled targeted backend safety alignment.

AI USE STATEMENT

Generative AI systems were used to support the research process and improve writing. Llama-Guard-4-12B, ShieldGemma-9B, Qwen3Guard-8B-Gen, and WildGuard-7B were used to generate the guardrail annotations employed in constructing the training data. GPT-5.5 was used to generate mutation inputs for the LLM-Fuzzer evaluation and as the automated harm judge used to compute attack success rates, as described in §4.1. Separately, generative AI tools were used to provide feedback on paper restructuring and LaTeX migration. All AI-assisted work was manually checked and adjusted by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies jailbreak prompts, adversarial samples, and harmful model outputs, which may raise dual-use concerns. We limit the study to computational experiments on public datasets, including PKU-SafeRLHF-30K, JailbreakHub, JailbreakBench, AdvBench, and BIG-Bench Hard, and do not collect private user data or involve human participants. We took several steps to reduce potential risks. Experiments on open-source models were conducted on independently operated servers in a controlled environment isolated from external users and production systems. For closed-source GPT-5.5 accessed through OpenAI's API, we followed the provider's service policies, respected rate limits, and avoided activities that could disrupt providers or other users. We also avoid releasing harmful model outputs generated during the experiments. As described in our Reproducibility statement, released artifacts are limited to sanitized code, aggregate results, reproducibility scripts, and non-actionable examples necessary to understand and reproduce the study. The broader goal of this work is to improve trustworthy AI deployment by reducing unsafe responses to jailbreak attacks while avoiding additional optimization pressure on explicit safety–utility conflicts through the helpfulness-gated Stubbornness Penalty.

REPRODUCIBILITY STATEMENT

During the double-blind review period, reviewers can access our anonymized artifact at `https://anonymous.4open.science/r/GuardAlign-8ACE/`. The method and objective are specified in §3.2 and §3.3; the datasets, models, hardware, optimization settings, and random seed are described in §4.1; the jailbreak-set construction and attack configurations are documented in §A; and the complete hyperparameter-selection protocol is reported in §B. The artifact provides the released implementations and configurations for GuardPO, G-SimPO, and G-DPO, pinned software dependencies, and training and evaluation scripts. The repository has been anonymized by removing author names, affiliations, account identifiers, and identifying metadata.

REFERENCES

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J Pappas, Florian Tramer, et al. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *Advances in Neural Information Processing Systems*, 37:55005–55029, 2024.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Yi Dong, Ronghui Mu, Yanghao Zhang, Siqi Sun, Tianle Zhang, Changshun Wu, Gaojie Jin, Yi Qi, Jinwei Hu, Jie Meng, et al. Safeguarding large language models: A survey. *Artificial intelligence review*, 58(12):382, 2025.

Google. Gemma-2-9b-it. https://huggingface.co/google/gemma-2-9b-it, 2026.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. *Advances in neural information processing systems*, 37:8093–8131, 2024.

Luxi He, Mengzhou Xia, and Peter Henderson. What's in your" safe" data?: Identifying benign data that breaks safety. In *ICLR 2024 Workshop on Navigating and Addressing Data Problems for Foundation Models*.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Josef Dai, Boren Zheng, Tianyi Alex Qiu, Jiayi Zhou, Kaile Wang, Boxun Li, et al. Pku-saferlhf: Towards multi-level safety alignment for llms with human preference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31983–32016, 2025.

Heegyu Kim, Sehyun Yuk, and Hyunsouk Cho. Break the breakout: Reinventing lm defense against jailbreak attacks with self-refinement. *arXiv preprint arXiv:2402.15180*, 2024.

Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. Drattack: Prompt decomposition and reconstruction makes powerful llm jailbreakers. *arXiv preprint arXiv:2402.16914*, 2024.

Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. Toxicchat: Unveiling hidden challenges of toxicity detection in real-world user-ai conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 4694–4702, 2023.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

Xiaogeng Liu, Peiran Li, G. Edward Suh, Yevgeniy Vorobeychik, Zhuoqing Mao, Somesh Jha, Patrick McDaniel, Huan Sun, Bo Li, and Chaowei Xiao. Autodan-turbo: A lifelong agent for strategy self-exploration to jailbreak llms. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*, 2025a.

Yue Liu, Hongcheng Gao, Shengfang Zhai, Jun Xia, Tianyi Wu, Zhiwei Xue, Yulin Chen, Kenji Kawaguchi, Jiaheng Zhang, and Bryan Hooi. Guardreasoner: Towards reasoning-based llm safeguards. *arXiv preprint arXiv:2501.18492*, 2025b.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

Meta. Llama-3.1-8b-instruct. https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct, 2026.

Anselm Paulus, Arman Zharmagambetov, Chuan Guo, Brandon Amos, and Yuandong Tian. Advprompter: Fast adaptive adversarial prompting for llms. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a85b405ed65c6477a4fe8302b5e06ce7-Paper-Conference.pdf.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pp. 1671–1685, 2024.

Minkyoo Song, Hanna Kim, Jaehan Kim, Seungwon Shin, and Sooel Son. Refusal is not an option: Unlearning safety alignment of large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 319–338, 2025.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

The Mistral AI Team. Mistral-7b-instruct-v0.3. `https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3`, 2024.

The Qwen AI Team. Qwen3-8b. `https://huggingface.co/Qwen/Qwen3-8B`, 2026.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Xunguang Wang, Zhenlan Ji, Wenxuan Wang, Zongjie Li, Daoyuan Wu, and Shuai Wang. Sok: Evaluating jailbreak guardrails for large language models. In *Proc. IEEE Symposium on Security and Privacy (S&P)*, 2026.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36:80079–80110, 2023.

Junkang Wu, Xue Wang, Zhengyi Yang, Jiancan Wu, Jinyang Gao, Bolin Ding, Xiang Wang, and Xiangnan He. Alphadpo: Adaptive reward margin for direct preference optimization. In *Forty-second International Conference on Machine Learning*.

Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. {LLM-Fuzzer}: Scaling assessment of large language model jailbreaks. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 4657–4674, 2024.

Wenjun Zeng, Yuchi Liu, Ryan Mullins, Ludovic Peran, Joe Fernandez, Hamza Harkous, Karthik Narasimhan, Drew Proud, Piyush Kumar, Bhaktipriya Radharapu, et al. Shieldgemma: Generative ai content moderation based on gemma. *arXiv preprint arXiv:2407.21772*, 2024.

Haiquan Zhao, Chenhan Yuan, Fei Huang, Xiaomeng Hu, Yichang Zhang, An Yang, Bowen Yu, Dayiheng Liu, Jingren Zhou, Junyang Lin, et al. Qwen3guard technical report. *arXiv preprint arXiv:2510.14276*, 2025.

Chujie Zheng, Ziqi Wang, Heng Ji, Minlie Huang, and Nanyun Peng. Weak-to-strong extrapolation expedites alignment. In *ICML 2024 Workshop on Models of Human Feedback for AI Alignment*.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

APPENDIX

A JAILBREAK TEST SET DETAILS

Table 3 summarizes the benchmark sources and attack configurations used to construct the jailbreak test sets for the safety evaluation in the main text. These test sets cover a manually curated attack and four automated attacks spanning optimization-based, generation-based, and implicit strategies.

Table 3: Benchmark sources and attack configurations used to construct the jailbreak test sets.

Dataset	Seed prompts	Jailbreak attack	Attack type	Evaluated prompts
JailbreakHub (Shen et al., 2024)	620	IJP (Shen et al., 2024)	Manual	620
JailbreakBench (Chao et al., 2024)	100	GCG (Zou et al., 2023)	Optimization-based	100
	100	AutoDAN (Liu et al., 2024)	Optimization-based	100
	100	AutoDAN-Turbo (Liu et al., 2025a)	Generation-based	100
	100	DrAttack (Li et al., 2024)	Implicit	100
AdvBench (Zou et al., 2023)	520	GCG	Optimization-based	520
	520	AutoDAN	Optimization-based	520
	520	AutoDAN-Turbo	Generation-based	520
	520	DrAttack	Implicit	520

Dataset Construction. From the 1,405 in-the-wild jailbreak prompts (IJP) collected by *JailbreakHub* (Shen et al., 2024), we use 620 prompts to match the size of each automated attack test set. For the automated test sets, we apply four complementary methods to the harmful seed prompts from *JailbreakBench* (Chao et al., 2024) and *AdvBench* (Zou et al., 2023): *GCG* (Zou et al., 2023) and *AutoDAN* (Liu et al., 2024) are optimization-based attacks, *AutoDAN-Turbo* (Liu et al., 2025a) is generation-based, and *DrAttack* (Li et al., 2024) is an implicit attack. Applying each attack to all 100 JailbreakBench seeds and all 520 AdvBench seeds produces the attack-specific test collections reported in the table.

Attack Configuration. For GCG, we optimize adversarial suffixes against the target LLM with a batch size of 512 for 500 iterations. For AutoDAN, we use the hierarchical genetic algorithm variant with a crossover probability of 0.5 and a mutation probability of 0.01, and run the optimization for 500 iterations. For AutoDAN-Turbo, we use the strategy library from the official implementation and run 150 epochs for each prompt. GCG, AutoDAN, and AutoDAN-Turbo use Llama-3.1-8B-Instruct (Meta, 2026) as the attack target. For DrAttack, we follow the original implementation and its official prompt template, using Llama-2-7B (Touvron et al., 2023) as the attack target.

B GuardPO Hyperparameter Search

We select the GuardPO-specific hyperparameters with a three-stage successive-halving procedure on Gemma-2-9B-IT. The search varies the Detection Penalty weight λ_1, the Stubbornness Penalty weight λ_2, the all-correct attenuation factor α, and the target normalized unsafe confidence τ_G. The two penalty weights remain strictly positive throughout hyperparameter selection, so every search candidate is a complete GuardPO configuration rather than a SimPO configuration. We allow $\alpha = 0$ to disable only the all-correct confidence-deficit branch; τ_G is inactive in this case. The complete discrete search ranges are $\lambda_1 \in \{0.10, 0.25, 0.50, 0.75, 1.00, 1.25, 1.50\}$, $\lambda_2 \in \{0.10, 0.25, 0.50, 0.75, 1.00, 1.50, 2.00\}$, $\alpha \in \{0, 0.10, 0.25, 0.40, 0.60, 0.75\}$, and $\tau_G \in \{0.35, 0.50, 0.65, 0.80, 0.90\}$ conditional on $\alpha > 0$.

All candidates use the same unsafe-prompt preference data $\mathcal{D}$, guardrail annotations, normalization statistics computed on $\mathcal{D}$, data order, and model initialization. We fix $\beta = 2.5$ and $\gamma_{\text{base}}/\beta = 0.55$, giving $\gamma_{\text{base}} = 1.375$, and use AdamW with a cosine schedule, learning rate 10^{-6}, 10% warmup, and seed 42. LoRA uses rank 64, scaling factor 64, and dropout 0.05. Hyperparameters are selected only on development data disjoint from $\mathcal{D}$ and the final evaluation data. A candidate is eligible only if all judge outputs are valid; eligible candidates are ranked by ASR, followed by the numbers of score-5 and score-4 responses, mean judge score, guardrail-missed-subset ASR, and helpfulness-agreement-subset ASR.

B.1 Stage I: Broad Space-Filling Search

Stage I trains the 24 configurations in Table 4 for one epoch from the same Gemma-2-9B-it initialization. The configurations form a deterministic balanced space-filling design over the prespecified ranges instead of a full $7 \times 7 \times 6 \times 5$ grid. The previously tested setting $(0.75, 1.00, 0.25, 0.65)$ is retained as S1-01 only as an ordinary observed point and is not used as the center of the design.

We evaluate all 24 adapters on the same development prompts and advance the eight highest-ranked eligible candidates.

Table 4: Stage-I GuardPO configurations. All candidates are trained for one epoch. A dash under τ_G marks the inactive threshold when $\alpha = 0$.

ID	λ_1	λ_2	α	τ_G	ASR (%)	ID	λ_1	λ_2	α	τ_G	ASR (%)
S1-01	0.75	1.00	0.25	0.65	8.00	S1-13	1.50	0.75	0.40	0.50	2.00
S1-02	0.10	0.50	0.75	0.80	5.00	S1-14	1.50	0.10	0.60	0.65	5.00
S1-03	0.25	0.25	0.60	0.90	2.00	S1-15	0.25	2.00	0.60	0.35	5.00
S1-04	1.00	2.00	0.40	0.50	3.00	S1-16	1.25	0.50	0.75	0.80	7.00
S1-05	0.25	1.50	0.10	0.35	4.00	S1-17	1.00	1.00	0.10	0.35	5.00
S1-06	1.50	2.00	0.00	–	8.00	S1-18	0.10	2.00	0.25	0.90	8.00
S1-07	1.25	1.50	0.60	0.65	6.00	S1-19	0.75	0.25	0.25	0.50	5.00
S1-08	0.50	0.75	0.00	–	7.00	S1-20	0.75	0.75	0.75	0.35	7.00
S1-09	1.25	1.50	0.25	0.80	7.00	S1-21	0.50	0.10	0.10	0.90	5.00
S1-10	1.00	0.25	0.40	0.80	4.00	S1-22	0.10	0.10	0.00	–	8.00
S1-11	0.50	1.00	0.75	0.90	11.00	S1-23	1.50	0.50	0.10	0.65	4.00
S1-12	0.75	0.10	0.40	0.50	5.00	S1-24	0.10	1.00	0.00	–	8.00

B.2 Stage II: Three-Epoch Continuation

Stage II resumes the eight selected Stage-I checkpoints and continues each run until it reaches three total training epochs. Resuming rather than restarting preserves the optimization trajectory used for successive halving, while the model initialization, data order, optimizer, and all hyperparameters remain unchanged. We then reevaluate the eight candidates on exactly the same development sets and with the same judge configuration. Table 5 records the Stage-I provenance and the three-epoch selection results; the three highest-ranked eligible candidates advance to Stage III.

Table 5: Stage-II continuation of the eight highest-ranked Stage-I candidates. Each candidate is resumed to three total epochs.

Stage-I rank	Source ID	λ_1	λ_2	α	τ_G	ASR (%)	Advance
1	S1-04	1.00	2.00	0.40	0.50	2.00	Yes
2	S1-10	1.00	0.25	0.40	0.80	2.00	Yes
3	S1-12	0.75	0.10	0.40	0.50	2.00	No
4	S1-24	0.10	1.00	0.00	–	1.00	Yes
5	S1-11	0.50	1.00	0.75	0.90	3.00	No
6	S1-13	1.50	0.75	0.40	0.50	3.00	No
7	S1-03	0.25	0.25	0.60	0.90	3.00	No
8	S1-23	1.50	0.50	0.10	0.65	3.00	No

B.3 Stage III: Five-Epoch Finalists

Stage III resumes the three Stage-II finalists and continues each run until it reaches five total epochs. We apply the same ranking rule one final time and select the values of $(\lambda_1, \lambda_2, \alpha, \tau_G)$ from the development results in Table 6. Only after this choice is frozen do we evaluate the selected configuration on the final test sets.

Table 6: Stage-III continuation of the three highest-ranked Stage-II candidates. Each candidate is resumed to five total epochs.

Stage-II rank	Source ID	λ_1	λ_2	α	τ_G	ASR (%)	Selected
1	S2-R04	0.10	1.00	0.00	–	3.00	No
2	S2-R01	1.00	2.00	0.40	0.50	0.00	Yes
3	S2-R02	1.00	0.25	0.40	0.80	2.00	No

Search outcome. At five total epochs, S2-R01 achieves the lowest development ASR (0.00%), compared with 3.00% for S2-R04 and 2.00% for S2-R02. We therefore select $(\lambda_1, \lambda_2, \alpha, \tau_G) = (1.00, 2.00, 0.40, 0.50)$; within the evaluated grid, the best setting combines a strong Detection Penalty weight and the largest tested Stubbornness Penalty weight with moderate all-correct attenuation.