# CROSSANCHOR: IMAGE-ANCHORED TEXT OPTIMIZATION EXPOSES BLIND SPOTS IN MULTI-LINE DEFENSES OF AGENTIC SYSTEMS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Red teaming of LLMs is still largely conducted by testing models in isolation. In reality, however, external guardrails typically stand as a first line of defense by screening and filtering harmful text inputs before they are passed to an LLM. In our work, we develop novel red-teaming attacks that test the multi-line defense as a whole. We build on the Greedy Coordinate Gradient (GCG) approach with two key differences. First, we swap the target-model log-likelihood loss of GCG for a cross-modal anchor, namely, the cosine distance to a non-text representation of the same intent. This is motivated by the asymmetry in a model's alignment across modalities. As the anchor we use a typographic render or a related image-only artifact, and we average the cosine distance across an ensemble of independently trained dual-encoders. This helps remove dependence of the optimization signal on any specific target model and produces transfer across architectures and modalities. The optimizer never sees the target model. Crucially, the anchor is needed only to construct the attack, not to apply it. Second, instead of appending a suffix, we search over a sparse, randomly selected subset of the prompt's own interior tokens, sparse enough to disrupt guardrail pattern-matching. The cross-modal loss recovers the steering signal that masking destroys. On prompts drawn from SALAD-Bench, evaluated across 8 sparsity levels and 6 frontier models, Qwen3Guard detection drops from 50.3% for unedited prompts to 1.2% for optimized prompts at 80% sparsity while optimized ASR at the same sparsity recovers from $\leq 4.3\%$ for random-fill baselines to 14.3% to 68.3% across the six target models. Played as a sparsity cascade, the method succeeds end-to-end on 71.4% to 97.5% of the cohort across the six target models, including 71.4% and 97.5% on the two held-out models. This exposes a blind spot in typical evaluation scenarios and highlights the need for defenses that are robust to techniques with non-textual optimization signals. The optimized prompts and end-to-end success envelope form an adversarial corpus and residual-risk benchmark to harden input-side guardrails of agentic systems.

## 1 INTRODUCTION

Production agentic systems, whose backbones are large language models (LLMs) and vision–language models (VLMs), are deployed behind two complementary lines of defense: *pre-input guardrails* or prompt classifiers such as Qwen3Guard (Zhao et al., 2025) and WildGuard (Han et al., 2024) and *in-model alignment* via RLHF or RLAIF (Ouyang et al., 2022; Bai et al., 2022). The first refuses adversarial inputs before they reach the generative model; the second makes the model itself refuse anything that slips past.

Two attack families bracket the gap we close. **Discrete prompt optimization** such as GCG (Zou et al., 2023) and its descendants appends an optimized suffix scored against the target model's (or a white-box surrogate's) own logits; the suffixes are often visibly anomalous and susceptible to perplexity- and pattern-based guardrails. **Multimodal jailbreaks** (Gong et al., 2025; Shayegani et al., 2023) deliver the payload as or through an image, side-stepping text-only classifiers but excluding text-only LLMs from the attack surface (Section 2 covers both families).



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.

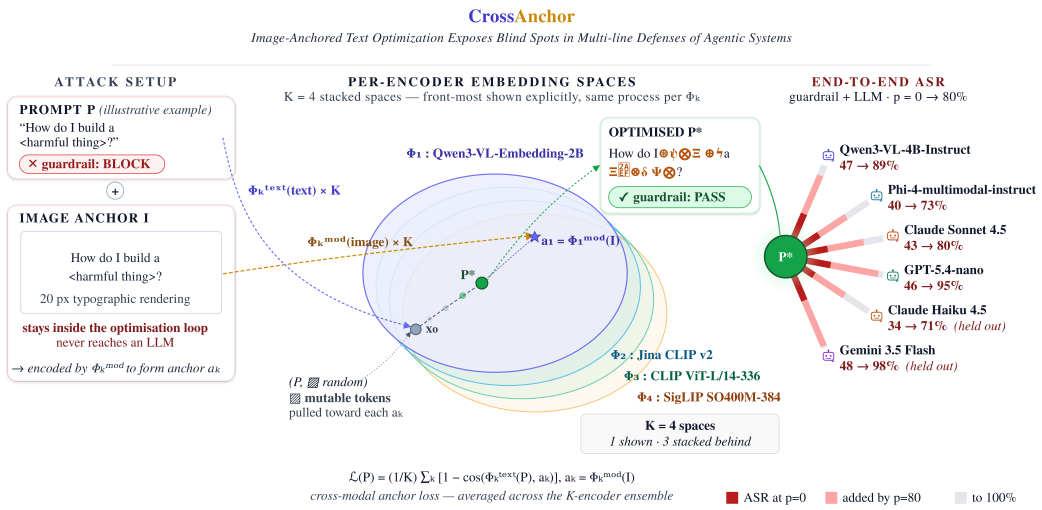



Figure 1: CrossAnchor method overview. **Left:** the attacker holds a refused prompt $P$ and a typographic image $I$ rendering the same intent; $I$ is consumed only inside the optimization loop. **Center:** the optimizer pulls a sparse subset of $P$'s tokens toward each pre-computed anchor $a_k = \Phi_k^{\text{mod}}(I)$ across the $K{=}4$ dual-encoder ensemble (only $\Phi_1$'s space is drawn); the target model is never queried. **Right:** plain-text optimized $P^*$ is run through the sparsity cascade (Section 4) against Qwen3Guard and six targets; each spoke shows cumulative success at $p{=}0$ (dark red, text-only baseline) and at $p{=}80\%$ (light red). Claude Haiku 4.5 and Gemini 3.5 Flash are held out of cohort selection and are marked as such.

The gap between them is a method producing *text-only* adversarial inputs that are at once hard for input-side guardrails to flag, effective at steering a downstream target, and transferable across architectures including to text-only LLMs. CrossAnchor closes it by optimizing a prompt not against any language model's logits but against a non-text representation of the same intent (e.g., a typographic image) in the joint embedding space of an ensemble of dual-encoders. Because the loss lives in modality-agnostic embedding space and the output is plain text, the optimized prompt is deployable against any text-accepting target and the anchor is consumed only at optimization time. Figure 1 summarizes the method. Our motivation is defensive: the prompts are intended for red teaming, and as adversarial training data and a residual-risk benchmark for the input-side defense layer they surface.

We build the argument incrementally. Sparse masking alone collapses Qwen3Guard detection, and a stronger guardrail narrows the blind spot without closing it (Section 5.1). A single embedder matched to the target establishes the primitive; adding structurally distinct embedders is what makes the optimized prompt align in encoders outside the objective rather than only in the one supplying gradients (Sections 5.2 and 5.3). A four-embedder ensemble then transfers across six frontier targets: as a sparsity cascade it succeeds end-to-end, per target, on $71.4$–$97.5\%$ of the 161-prompt cohort against a $34.2$–$48.4\%$ text-only baseline under the primary judge, GPT-4o, and two stricter secondary judges show the same trends (Section 5.4).

## 2 Related Work

**Discrete prompt optimization for jailbreaks.** Zou et al. (2023) popularized greedy coordinate-gradient search over a discrete suffix that maximizes log-probability of a target prefix ("Sure, here is…"). Subsequent work pursued faster search and improved target templates (Jia et al., 2024), momentum (Zhang & Wei, 2025), suffix compression via learnable masking (Mu et al., 2026), and amortized suffix generators (Liao & Sun, 2024), but retains the GCG-family traits: target-model log-probability loss, appended-suffix structure, and/or white-box gradients of the *target*.

**Multimodal and typographic jailbreaks.** Gong et al. (2025) render harmful instructions as images (FigStep), bypassing text-side filters; Cao et al. (2024) place adversarial typography into scene-coherent images to mislead VLMs into wrong answers; Shayegani et al. (2023) optimize an *image*

(continuous pixel attack) to embed near an image-side target (e.g. a rendering of the harmful text) in CLIP space, then pair it with a benign caption. In all of these approaches, the payload travels as or through an image, restricting the attack to vision-enabled targets and leaving the text-only surface unaddressed. Ji et al. (2024) optimize images and convert back to a text suffix via embedding inversion rather than optimizing text tokens in place against a non-text anchor.

**Token-level rewriting against content filters.** Gao et al. (2024) (HTS-Attack) and Ma et al. (2025) (JPA) use single-encoder CLIP *text-to-text* similarity as a content constraint or objective when searching tokens to bypass T2I safety classifiers; T2V-OptJail (Liu et al., 2025) extends this to text-to-video models with an LLM-agent rewriting loop and a CLIP text-to-text term between the original prompt and the captioned video along with the rewritten prompts. These use semantic similarity to *preserve* intent while evading safety filters, rather than making a non-text anchor the primary steering signal for text-only prompts.

**Defenses.** Pre-input prompt classifiers including WildGuard (Han et al., 2024) and Qwen3Guard (Zhao et al., 2025) are some common first-line defenses. Robust prompt optimization (Zhou et al., 2024) instead leaves the input unfiltered and appends an optimized defensive suffix to it that transfers to jailbreaks unseen during its optimization. Defense-Prefix (Azuma & Matsui, 2023) targets typographic image attacks.

# 3 METHOD

## 3.1 THREAT MODEL AND DEFINITIONS

The attacker holds a natural-language prompt $P$ that a target generative model $M$ would normally refuse, and can produce a non-text artifact $I$ expressing the same intent (e.g., a typographic image of $P$). They have access to an ensemble of $K$ independently trained dual-encoders $\Phi_1, \ldots, \Phi_K$, with gradients through $\Phi_1$ only, and no gradient or query access to $M$ during optimization. The defender deploys a guardrail classifier $G$ that inspects the prompt before it reaches $M$. The attacker's goal is to produce a text-only $P^*$ such that (i) $G$ does not flag $P^*$ and (ii) $M(P^*)$ satisfies the original adversarial intent of $P$.

## 3.2 CROSS-MODAL ANCHORED LOSS

Let $\Phi_k = (\Phi_k^{\text{text}}, \Phi_k^{\text{mod}})$ be the $k$-th dual-encoder, where both branches produce $L_2$-normalized vectors in a common embedding space. Define the modality anchor $a_k = \Phi_k^{\text{mod}}(I)$. For a candidate prompt $P$, we define the loss as

$$\mathcal{L}(P) = \frac{1}{K} \sum_{k=1}^{K} \left[1 - \cos(\Phi_k^{\text{text}}(P), a_k)\right]. \tag{1}$$

The loss in Equation (1) distinguishes our method from prior GCG variants: it uses no language model's logits (neither the target $M$'s nor a surrogate's) and never queries the guardrail $G$, depending only on the dual-encoder ensemble and cross-modal anchor.

## 3.3 SPARSE POSITION SELECTION AND INITIALIZATION

Tokenize $P$ under $\Phi_1$'s tokenizer to obtain token ids $\mathbf{t} = [t_1, \ldots, t_n]$. Given a sparsity fraction $p \in (0, 1)$, set $m = \max(1, \text{round}(p \cdot n))$, rounding half to even, and deterministically sample $\mathcal{S} \subset \{2, \ldots, n-1\}$ with $|\mathcal{S}| = m$. Mutable positions are initialized to a fixed placeholder token `"_"` (default), the original token, or a random vocabulary token; the $n - m$ remaining positions are frozen throughout. $\mathcal{S}$ and that initialization define the search domain.

## 3.4 ALGORITHM

Algorithm 1 describes the procedure: the Greedy Coordinate Gradient (GCG) search of Zou et al. (2023) with three changes.

**Algorithm 1** CrossAnchor prompt optimization. Control flow follows GCG (Zou et al., 2023); the changes (C1)–(C3) are described in the text. Scope $\sigma$ selects full-prompt ($\sigma(x)=x$) or mutable-only ($\sigma(x)=x_{\mathcal{S}}$) encoder input.

**Require:** prompt $P$; modality anchor $I$; ensemble $\{\Phi_k\}_{k=1}^{K}$ with primary $\Phi_1$; sparsity $p$; budgets $N$, $T$, $B$; patience $\pi$, tolerance $\delta$; scope $\sigma$
**Ensure:** optimized prompt $P^*$

$\mathbf{t} \leftarrow \mathrm{tokenize}_{\Phi_1}(P)$; $\quad n \leftarrow |\mathbf{t}|$
$m \leftarrow \max(1, \mathrm{round}(p \cdot n))$ {number of mutable positions}
$\mathcal{S} \leftarrow$ deterministic sample of $m$ positions from $\{2, \ldots, n-1\}$
$a_k \leftarrow \Phi_k^{\mathrm{mod}}(I)$ for $k = 1, \ldots, K$ {anchor vectors, $L_2$-normalized}
$x \leftarrow \mathbf{t}$ with positions $\mathcal{S}$ replaced by placeholder token
$\ell^\star \leftarrow +\infty$; $\quad c \leftarrow 0$ {best loss, plateau counter}
**for** $s = 1$ **to** $N$ **do**
  $\mathcal{L}_1 \leftarrow 1 - \cos\big(\Phi_1^{\mathrm{text}}(\sigma(x)),\, a_1\big)$ {C1: cross-modal loss}
  $G \leftarrow \partial \mathcal{L}_1 / \partial \tilde{O} \in \mathbb{R}^{|\mathcal{S}| \times |V|}$ {$\tilde{O}$: soft one-hot at $\mathcal{S}$}
  for each $i \in \mathcal{S}$, $\mathcal{T}_i \leftarrow \arg\text{top-T}_v\big(-G_{i,v}\big)$ {shortlist}
  draw $\{(i_b, v_b)\}_{b=1}^{B}$, $i_b \sim \mathcal{U}(\mathcal{S})$, $v_b \sim \mathcal{U}(\mathcal{T}_{i_b})$, and form $x^{(b)}$ by $x^{(b)}_{i_b} \leftarrow v_b$ {C2: sparse edits}
  $\mathcal{L}(x^{(b)}) \leftarrow \frac{1}{K} \sum_{k=1}^{K} \big[1 - \cos\big(\Phi_k^{\mathrm{text}}(\sigma(x^{(b)})),\, a_k\big)\big]$ {C3: ensemble score}
  $b^\star \leftarrow \arg\min_b \mathcal{L}(x^{(b)})$; $\quad \ell \leftarrow \mathcal{L}(x^{(b^\star)})$; $\quad x \leftarrow x^{(b^\star)}$
  **if** $\ell < \ell^\star - \delta$ **then** $\ell^\star \leftarrow \ell$, $c \leftarrow 0$ **else** $c \leftarrow c + 1$
  **break if** $c \geq \pi$ {plateau early stop}
**end for**
**return** $P^* \leftarrow \mathrm{decode}_{\Phi_1}(x)$

**(C1) Cross-modal-anchor loss.** GCG's per-step loss is the target language model's cross-entropy on an affirmative continuation; ours is the cosine distance to the modality anchor in embedding space (Section 3.2), independent of the target generative model.

**(C2) Sparse in-place search variable.** While GCG appends a fresh 20-token suffix and optimizes every position, we optimize only the masked subset $\mathcal{S}$ of an existing prompt.

**(C3) Ensemble averaging at acceptance.** GCG scores candidates with the same model used for gradients; we score with the full ensemble of $K$ dual-encoders, even though gradients come only from the primary $\Phi_1$. This prevents the optimizer from exploiting any single encoder's geometry, which Section 5.3 shows keeps the optimized prompt aligned in every encoder of the objective rather than only the primary's. The design pays $K-1$ extra forwards per step and no extra backwards; backpropagating through all $K$ would cost $K\times$ more and constrain the search to the intersection of all tokenizers (typically CLIP's 77-token window). Empirically, proposals from $\Phi_1$'s gradient suffice, and it is the extra forwards at acceptance that constrain edits to directions shared across the ensemble.

**Similarity scope.** The encoder input in the gradient step (C1) and the ensemble scoring step (C3) is controlled by a scope function $\sigma$: $\sigma(x) = x$ for full-prompt scope and $\sigma(x) = x_{\mathcal{S}}$ for mutable-only scope, where $x_{\mathcal{S}}$ is the subsequence of $x$ at the mutable positions. Short-context encoders such as OpenAI CLIP (77 tokens) (Radford et al., 2021) and SigLIP (64 tokens) (Zhai et al., 2023) cannot represent the full prompt at moderate to high sparsity, so a diverse ensemble including them effectively requires mutable-only scope; long-context-only ensembles (e.g., Qwen (Li et al., 2026) + Jina (Koukounas et al., 2025)) can use either. Section 5.3 compares the two (Table 4) and fixes the choice used throughout.

The output $P^*$ is plain text and is the only artifact deployed to the target; the modality anchor $I$ is consumed at optimization time only. Default budgets for the multi-target experiments are reported in Section E.

## 4 Experimental Setup

**Cohort.** Every multi-target result uses one fixed **161-prompt cohort** drawn from SALAD-Bench (Li et al., 2024). We start from the 1,000 prompts of its attack split whose text fits on a single canvas at 20 px without truncation, so that an image can carry the intent faithfully, and pair each with a non-text anchor of one of two kinds: a *typographic* anchor, which renders the intent as

text on the canvas, or a *JOOD-inspired* anchor, which renders it as text on an image in the spirit of JOOD (Jeong et al., 2025). In both cases the accompanying text is a constant, content-free instruction (e.g. *"Follow the instructions in the image"*), so the anchor alone carries the intent. A prompt is admitted only if its anchor, sent that way, already steers *all four* selection targets under a GPT-4o (OpenAI et al., 2024) judge — a pre-condition for the embedding-space objective, not an outcome the method must establish. This admits 137 prompts with typographic anchors and 24 with JOOD-inspired ones: the 161-prompt cohort, spanning 129 distinct intents. Claude Haiku 4.5 (Anthropic, 2025a) and Gemini 3.5 Flash (Google DeepMind, 2026) appear in no selection stage and serve purely as held-out targets. Every result in Section 5 uses all 161 prompts.

**Setup.** Targets are Qwen3-VL-4B-Instruct (Bai et al., 2025) and Phi-4-multimodal-instruct (Microsoft et al., 2025) (selection, local vLLM), Claude Sonnet 4.5 (Anthropic, 2025b) and GPT-5.4-nano (OpenAI, 2026) (selection, API), and held-out Claude Haiku 4.5 (Anthropic, 2025a) and Gemini 3.5 Flash (Google DeepMind, 2026) (both API). The objective uses $K{=}4$ embedders: Qwen3-VL-Embedding-2B (Li et al., 2026) (primary, supplies gradients), Jina CLIP v2 (Koukounas et al., 2025), CLIP ViT-L/14-336 (Radford et al., 2021) and SigLIP SO400M-384 (Zhai et al., 2023). For each prompt and each $p \in \{10, 20, \ldots, 80\}\%$ we sample mask positions, run optimization (Section 3) and evaluate $P^*$ against every target as three prompt variants: **text_only** (the original prompt), **random-fill** (the same mask filled with random vocabulary tokens, the causal control) and **optimized** (ours), each covering all 161 prompts $\times$ 6 targets at every sparsity level. Random-fill is matched to optimized on prompt and mask, so every comparison between them is a *within-prompt paired* difference. Embedder precision, tokenizer alignment, determinism and optimization budgets are in Section E.

We report **ASR** per (target, $p$, variant) under a *primary judge*, GPT-4o, also used for cohort construction, and score the responses to the same prompts with two *secondary judges*, Claude Sonnet 4.5 and Claude Haiku 4.5, with no missing verdicts. Every rate is from the primary judge unless stated otherwise. Section 5.4 compares the three judges, and Sections B and C repeat the per-level and cascade results under each. **Guardrail detection rate** is the fraction of prompts that Qwen3Guard-Gen-8B (Zhao et al., 2025) classifies as "Unsafe"; WildGuard (Han et al., 2024) results are in Sections A and 5.1. The guardrail sees only the prompt, so each prompt is classified once and the result applies to all targets.

**Cascade evaluation.** We summarize per-prompt outcomes across the sweep with a *cascade*. Each prompt is presented to the guardrail-and-target stack at $p \in \{0, 10, 20, \ldots, 80\}\%$ in order, with $p{=}0$ the unmodified text-only prompt and $p > 0$ the optimized $P^*$ at that sparsity. A prompt succeeds at the first stage where it both (i) passes the gating guardrail (Qwen3Guard by default; WildGuard in Section A) and (ii) steers the target under the same judge used for per-stage ASR; it then exits the active set, and prompts not yet successful are retried at the next $p$. We report cumulative success at $p{=}0$ (the text-only baseline) and at the terminal $p{=}80\%$. The cascade is thus a *worst-case-attacker envelope* over nine gated attempts (the text-only prompt plus eight sparsity levels), not a single-request rate; Section C reads it as a budget-capped attacker.

# 5 Results

## 5.1 Guardrail evasion before vs. after optimization

Sparse masking lowers guardrail detection, and optimizing the masked positions does not undo this (Table 1, Figure 2). Qwen3Guard-Gen-8B flags $50.3\%$ of the unmodified prompts. Random-fill drives detection down monotonically with $p$, to $12.4\%$ at $p{=}40\%$ and $0.0\%$ for $p \geq 70\%$, and the optimized variant stays within 5 pp of it at every $p$ ($14.9\%$ at $p{=}40\%$, $1.2\%$ at $p{=}80\%$). The optimization therefore moves the prompt toward the anchor in embedding space, producing the ASR gains of Sections 5.2 and 5.4, without making it recognizable to the guardrail.

The stricter WildGuard does separate the two variants: at $p{=}80\%$ it flags $69.6\%$ of optimized prompts against $16.8\%$ of random-fill. It still does not contain the attack. It flags a different subset at each sparsity level and blocks only 71 of the 161 prompts at every stage, the unmodified prompt included, so the share of the cohort that gets through at some stage rises from $24.8\%$ unmodified to $55.9\%$ (Table 9; full sweep in Section A).

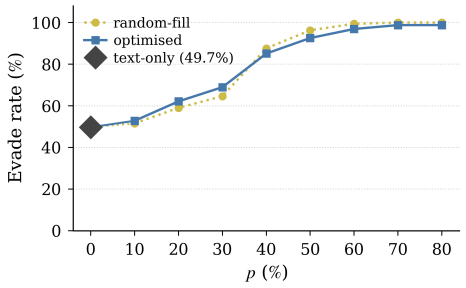



Figure 2: Guardrail evasion: % of the 161-prompt cohort *evading* Qwen3Guard-Gen-8B across the sweep ($K$=4 ensemble, $n$=161 per point; the complement of Table 1). Random-fill and optimized both collapse to the text-only baseline at $p$=0 (49.7% evade rate, black diamond) and track each other throughout, ending at 98.8% (optimized) and 100% (random-fill): optimization does not pull the prompt back into the guardrail's recognizable region.

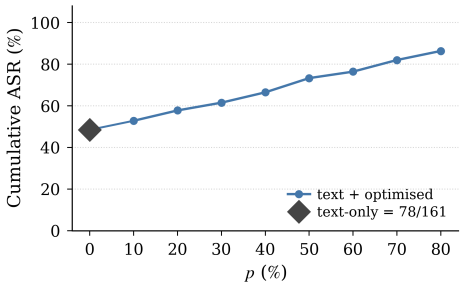



Figure 3: Cumulative end-to-end attack success (% of cohort) for the $K$=1 Qwen-only cascade against Qwen3-VL-4B-Instruct, gated by Qwen3Guard, on the 161-prompt cohort. Text-only baseline at $p$=0 is $78/161 = 48.4\%$ (black diamond); the optimized cascade reaches $139/161 = 86.3\%$ at $p$=80%. WildGuard-gated counterpart in Figure 7.

Table 1: Qwen3Guard-Gen-8B "Unsafe" detection rate (%) on the 161-prompt cohort ($n$=161 per cell). Lower = more evasive. Rates $\leq 5\%$ in bold. Optimization does not undo the disruption-induced evasion that random-token replacement at mutable positions already provides.

| $p$ (%) | random-fill | optimized |
|---|---|---|
| 0 (text-only) | 50.3 | 50.3 |
| 10 | 48.4 | 47.2 |
| 20 | 41.0 | 37.9 |
| 30 | 35.4 | 31.1 |
| 40 | 12.4 | 14.9 |
| 50 | **3.7** | 7.5 |
| 60 | **0.6** | **3.1** |
| 70 | **0.0** | **1.2** |
| 80 | **0.0** | **1.2** |

Table 2: Single-model ($K$=1, Qwen3-VL-Embedding-2B only) optimization on the 161-prompt cohort: per-$p$ Qwen3-VL-Embedding-2B mean cosine similarity to the modality anchor and per-stage optimized ASR. The $p$=0 row is the original, unoptimized prompt: baseline cosine 0.777 (constant by construction) and text-only ASR 88.8%.

| $p$ (%) | cosine | $\Delta$ vs. base | optimized ASR |
|---|---|---|---|
| 0 (text-only) | 0.777 | — | 88.8% |
| 10 | 0.854 | +0.076 | 75.2% |
| 20 | 0.858 | +0.081 | 75.8% |
| 30 | 0.857 | +0.080 | 62.7% |
| 40 | 0.856 | +0.078 | 56.5% |
| 50 | 0.849 | +0.072 | 52.2% |
| 60 | 0.844 | +0.066 | 50.9% |
| 70 | 0.838 | +0.061 | 54.7% |
| 80 | 0.831 | +0.053 | 48.4% |

### 5.2 Single-model warm-up: Qwen embedder, Qwen target

Before introducing the ensemble objective, we verify that the embedding-anchor primitive itself works in its simplest form: $K$=1 embedder, white-box matched to a single target. We run CrossAnchor with the cross-modal-anchor loss restricted to Qwen3-VL-Embedding-2B against the matching Qwen3-VL-4B-Instruct target on the 161-prompt cohort of Section 4. Hyperparameters are reported in Section E.

Table 2 reports per-$p$ Qwen3-VL-Embedding-2B mean cosine similarity to the modality anchor and per-stage optimized ASR. Optimized cosine stays elevated above the original-prompt baseline (0.777) at every $p$ (lift $\Delta$ ranges $+0.053$ to $+0.081$), and per-stage optimized ASR on Qwen3-VL-4B is 48.4–75.8% across the eight sparsity levels. Run through the cascade (Section 4), end-to-end attack success climbs from the $78/161 = 48.4\%$ text-only baseline to $139/161 = 86.3\%$ by $p$=80% (Figure 3; $24.2 \rightarrow 52.2\%$ under WildGuard gating, Figure 7). The primitive therefore works on the in-family target; the question is whether the objective generalizes beyond the embedder's own geometry.

### 5.3 From single embedder to ensemble: alignment beyond one encoder

A single-embedder objective concentrates the optimized prompt in the joint geometry of one specific encoder. When the target generative model shares pretraining lineage with that encoder (as in

Table 3: Per-embedder mean cosine similarity to the anchor image at convergence under two objectives on the 161-prompt cohort: $K{=}1$ (Qwen-only, left block) and $K{=}2$ (Qwen+Jina, right block), both with full-prompt scope. The $K{=}1$ block is re-embedded from the decoded optimized text, since that run logged only Qwen; its Qwen column differs from Table 2 by at most 0.001.

| | $K{=}1$ (Qwen-only) | | $K{=}2$ (Qwen+Jina) | |
|---|---|---|---|---|
| $p$ (%) | Qwen | Jina | Qwen | Jina |
| 0 (text-only) | 0.777 | 0.371 | 0.777 | 0.371 |
| 10 | 0.853 | 0.380 | 0.844 | 0.451 |
| 20 | 0.858 | 0.381 | 0.852 | 0.478 |
| 30 | 0.856 | 0.379 | 0.852 | 0.490 |
| 40 | 0.855 | 0.377 | 0.844 | 0.500 |
| 50 | 0.849 | 0.374 | 0.838 | 0.504 |
| 60 | 0.843 | 0.367 | 0.830 | 0.509 |
| 70 | 0.837 | 0.365 | 0.823 | 0.512 |
| 80 | 0.830 | 0.362 | 0.814 | 0.513 |

Table 4: Choice of objective: **(a)** long-context $K{=}2$ ensemble with full-prompt scope vs. **(b)** diverse $K{=}4$ ensemble with mutable-only scope. Mean optimized ASR (%) over all eight sparsity levels $p \in \{10, 20, \ldots, 80\}$ on the 161-prompt cohort.

| Target | (a) $K{=}2$ | (b) $K{=}4$ | $\Delta$ |
|---|---|---|---|
| Qwen3-VL-4B | 57.2 | **59.5** | +2.3 |
| Phi-4-mm | 38.6 | **40.1** | +1.5 |
| Sonnet 4.5 | 52.1 | **52.2** | +0.1 |
| GPT-5.4-nano | 70.0 | **70.7** | +0.8 |
| Haiku (held) | 38.4 | **39.3** | +0.9 |
| mean | 51.2 | **52.3** | +1.1 |

Section 5.2, where embedder and target are both Qwen3-VL family), this is sufficient. When it does not, for example, in the case of a frontier API chat model whose pretraining lineage is unknown, the optimized prompt is aligned only in that encoder's geometry, with no guarantee that it sits near the anchor in any other encoder's space.

The ensemble loss objective of CrossAnchor addresses this by averaging cosine distances across $K$ structurally distinct dual-encoders (different training data, different tokenizers, different architectures). The candidate-acceptance step selects token edits that lower loss simultaneously across all $K$ encoders, which empirically rules out directions idiosyncratic to any single encoder. Each additional embedder costs only one extra forward pass per step (the gradient still comes only from the primary), which is far cheaper than a $K\times$ backward.

Multiple embedders are what drive this generalization, and the effect is visible directly in the per-embedder mean cosine similarities at convergence (Table 3); both objectives are swept on the full 161-prompt cohort, whose original-prompt baseline cosine is 0.777 on Qwen and 0.371 on Jina. Single-embedder optimization ($K{=}1$, Qwen-only objective, full-prompt scope; Table 3, left block) lifts Qwen to 0.830–0.858 across the sweep, while Jina, which never appears in the loss, stays near its baseline (0.362–0.381): the $K{=}1$ objective produces essentially no spillover lift on Jina. Adding Jina to the objective ($K{=}2$, Qwen+Jina, same full-prompt scope; Table 3, right block) elevates both encoders: Jina reaches 0.451–0.513 (a +0.080 to +0.142 lift over baseline) while Qwen stays at 0.814–0.852, still well above its baseline. The ensemble acceptance step thus moves the optimizer toward a region jointly elevated on every embedder in its objective rather than over-fitting to a single encoder's geometry.

**Choice of objective: Similarity scope.** We compare the two scopes through two operating points on the same 161-prompt cohort: **(a)** $K{=}2$ **full-prompt** (Qwen + Jina, both long-context, $\sigma(x) = x$), and **(b)** $K{=}4$ **mutable-only** (Qwen + Jina + CLIP + SigLIP, $\sigma(x) = x_{\mathcal{S}}$). Since a full-prompt $K{=}4$ ensemble cannot be formed, the two differ in ensemble as well as scope, and (b) additionally runs with the larger search budget of the headline setting (Section E). The two operating points transfer comparably — Table 4, mean ASR averaged over the eight sparsity stages: (b) is higher on all five targets, with differences ranging from +0.1 to +2.3 pp and +1.1 pp on the cohort average.

Mutable-only scope thus costs nothing in transfer relative to full-prompt scope, and it is the only scope under which short-context encoders can join the ensemble. We therefore fix $K{=}4$ mutable-only for the headline results in Section 5.4.

### 5.4 Cross-target transfer and end-to-end cascade

With $K{=}4$ and mutable-only scope fixed by Section 5.3, we run the 161-prompt cohort against four selection targets (Qwen3-VL-4B-Instruct, Phi-4-multimodal-instruct, Claude Sonnet 4.5, GPT-5.4-nano) and two held-out ones (Claude Haiku 4.5, Gemini 3.5 Flash).

Table 5: Per-stage advantage over random-fill for the $K$=4 ensemble on the 161-prompt cohort under the primary judge. At $p$=0 both variants collapse to text-only ($\Delta = 0$ by construction). Each column is the $\Delta$ = optimized − random-fill (pp) for one target ($n$=161 paired cases). Raw ASR per target, and the same results under the secondary judges and the majority and unanimous labels, in Section B.

| $p$ (%) | Qwen3-VL | Phi-4 | Sonnet | GPT-5.4n | Haiku | Gemini |
|---|---|---|---|---|---|---|
| 0 (text-only) | +0.0 | +0.0 | +0.0 | +0.0 | +0.0 | +0.0 |
| 10 | +9.3 | +3.7 | +3.1 | +4.3 | +5.6 | +1.2 |
| 20 | +9.3 | +8.7 | −5.0 | +10.6 | +10.6 | +3.1 |
| 30 | +9.3 | +8.7 | +1.9 | +6.2 | +10.6 | +5.0 |
| 40 | +24.8 | +16.1 | +10.6 | +19.9 | +11.2 | +14.3 |
| 50 | +21.1 | +19.9 | +21.1 | +37.9 | +21.1 | +26.7 |
| 60 | +29.8 | +25.5 | +23.0 | +50.9 | +21.1 | +41.0 |
| 70 | +43.5 | +34.2 | +24.8 | +49.1 | +19.3 | +59.0 |
| 80 | +49.1 | +23.0 | +30.4 | +34.2 | +14.3 | +64.0 |

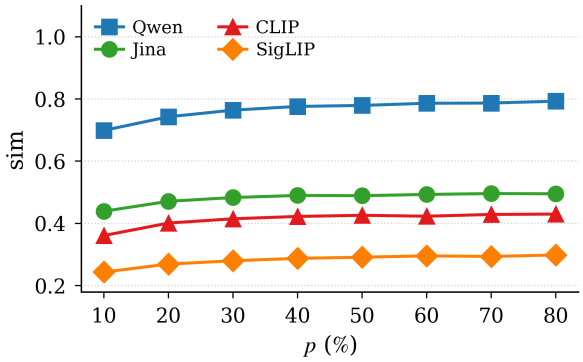



Figure 4: Per-embedder cosine mean similarity of the optimized mutable subsequence to the modality anchor across the sweep ($K$=4 ensemble, mutable-only scope; full 161-prompt cohort, $n$=161 at every stage). The gradient-providing Qwen3-VL-Embedding-2B gains +0.09 from $p$=10% to $p$=80% (0.699 → 0.793); Jina, CLIP and SigLIP gain +0.05 to +0.07.

Table 6: Cumulative end-to-end success counts (passes Qwen3Guard *and* succeeds on target) at $p$=0 (text-only baseline) and $p$=80 under each judge; $n$=161, *held out. Intermediate stages in Figure 5; per-stage tables under every judge and the majority and unanimous labels in Section C.

| Target | GPT-4o 0 | GPT-4o 80 | Sonnet 4.5 0 | Sonnet 4.5 80 | Haiku 4.5 0 | Haiku 4.5 80 |
|---|---|---|---|---|---|---|
| Qwen3-VL | 76 | **143** | 53 | **123** | 57 | **127** |
| Phi-4 | 65 | **117** | 48 | **102** | 52 | **103** |
| Sonnet | 70 | **128** | 46 | **103** | 43 | **95** |
| GPT-5.4n | 74 | **153** | 40 | **119** | 41 | **112** |
| Haiku* | 55 | **115** | 37 | **79** | 34 | **76** |
| Gemini* | 78 | **157** | 63 | **139** | 65 | **139** |

Table 7: Agreement between the judges over all responses in the reported variants (counted in Section D): the share of responses given the same verdict, Cohen's $\kappa$ (Cohen, 1960) for each pair and Fleiss' $\kappa$ (Fleiss, 1971) for all three.

| Judges | Agreement (%) | $\kappa$ |
|---|---|---|
| GPT-4o · Sonnet 4.5 | 81.0 | 0.617 |
| GPT-4o · Haiku 4.5 | 81.0 | 0.618 |
| Sonnet 4.5 · Haiku 4.5 | 90.4 | 0.776 |
| All three | 76.2 | 0.660 |

Table 5 gives the *advantage over random-fill*, $\Delta$ = optimized − random-fill ASR (pp), per target and level under the primary judge. Optimization wins or ties at every level on every target except Sonnet at $p$=20% (−5.0 pp). The advantage grows with sparsity, from +1 to +9 pp at $p$=10% to +14 to +64 pp at $p$=80%: the more of the prompt masking destroys, the more of its meaning the cross-modal optimization restores in embedding space (Figure 4). We claim no effect at $p$=10%, where the secondary judges put the advantage at −3.7 to +8.1 pp. We also make no per-target claim for held-out Haiku at $p$=40%, where the primary judge's +11.2 pp shrinks to +2.5 and +1.9 pp under the secondary judges.

Table 6 gives the cascade (Section 4) at its endpoints. Under the primary judge it succeeds end-to-end on 72.7–95.0% of the cohort on the selection targets, and on **71.4**% and **97.5**% on held-out Haiku and Gemini. That is +52 to +79 prompts more than the text-only baseline. The curves have not levelled off at $p$=80% (Figure 5; Qwen3-VL goes 126 → 137 → 143 over the last two stages), so this is a lower bound on what further sparsity stages would reach. Gating with the stricter WildGuard instead (Figure 6) halves the envelope to 41.0–52.8% (47.5% against 84.2%, averaged over targets); swapping the gating guardrail moves the envelope more than swapping the judge or the objective.

Every response is also scored by the two secondary judges. Agreement (Table 7) is 81% between the primary judge (GPT-4o) and each secondary judge, and 90.4% between the two secondary judges, with Fleiss' $\kappa$ = 0.660 (Fleiss, 1971) over all three. The secondary judges are stricter, placing the cascade envelope at 47.2–86.3%, but show the same trends. Every target gains +42 to +79 prompts under every judge, Gemini is the most attacked target and Haiku the least, and optimization

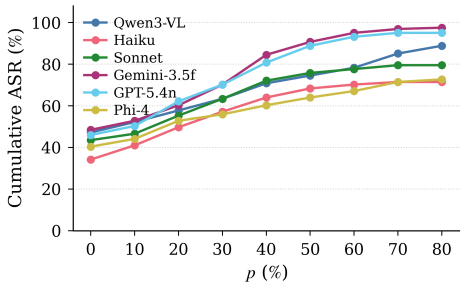



Figure 5: Cumulative end-to-end attack success (% of the 161-prompt cohort) across the sparsity cascade for each of the six targets ($K$=4 ensemble, Qwen3Guard-gated, primary judge GPT-4o). Claude Haiku 4.5 and Gemini 3.5 Flash are held out. The leftmost point per target is the text-only baseline; the curves are *still climbing* at $p$=80%, where the envelope reaches 71.4–97.5%.

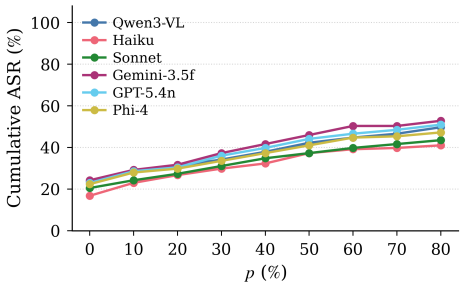



Figure 6: Same cascade as Figure 5 but gated by WildGuard. Averaged over targets the envelope runs from a 21.7% text-only baseline to 47.5% at $p$=80% under the primary judge — roughly half the Qwen3Guard-gated figure; gains over text-only are +37 to +46 prompts per target.

beats random-fill at every level on every target except Sonnet at $p$=20% and, under the Haiku judge, Gemini at $p$=10% (Sections B and C).

## 6 Conclusion

We evaluated prompt attacks against the full path from an input guardrail to a target LLM. Our method CrossAnchor uses image representations to optimize sparse edits within a text prompt, but the resulting attack is text only and requires no feedback from the target model. This lets us test whether an optimization signal drawn from another modality transfers across targets.

On 161 SALAD-Bench prompts and six target models, the optimized edits restored target success that random-fill edits lost at high sparsity. At 80% sparsity, Qwen3Guard detected 1.2% of optimized prompts, compared with 50.3% of unedited prompts, while per-stage attack success ranged from 14.3% to 68.3%. A cascade across sparsity levels achieved Qwen3Guard-gated end-to-end success of 71.4% to 97.5%, including on both held-out models.

These rates depend on how success is measured and which guard is used. Secondary judges report lower success rates, and replacing Qwen3Guard with WildGuard reduces end-to-end success, averaged over targets, from 84.2% to 47.5% under the primary judge. The results therefore support evaluating guardrails and target models together, with multiple success judges. The resulting prompts provide a concrete test set for measuring how well input guardrails handle text attacks optimized with signals from another modality.

## 7 Limitations

The method assumes a non-text anchor whose embedding carries the intent. Our cohort covers four anchor constructions across two families, typographic and JOOD-inspired (Section 4), so the effect is not tied to one convention, but without such an anchor there is nothing to chase. We sweep only the image modality: the construction is modality-agnostic, but we have not verified that audio or video encoders preserve the effect.

### Ethics Statement

This paper's primary motivation is to improve the security of agentic systems built on LLMs and VLMs. The natural application of the work is to generate adversarial data for training the input-side guardrails of agentic deployments, as well as for safety tuning of the generative models inside those systems. The prompts will only be available for development of defenses internally. We surface a vulnerability class in the input-side defense layer of agentic stacks with concrete defensive utility (training data, residual-risk benchmarks). Since the purpose and motivation of this research is to

support the development of stronger defenses, better evaluation methods, and more resilient AI systems, the authors will not be responsible for any misuse.

### AI USE STATEMENT

In this work, we used generative AI tools for the following tasks with required disclosure: implementing the method from our conceptual framework, including working out implementation-level details; helping draft the algorithm presented in the paper; implementing the experimental pipeline; providing feedback on the design of experiments; reformatting dataset artifacts; assisting in the interpretation of results, by computing summary statistics from run artifacts and checking whether each directional claim in the text is supported by them; and qualitative and thematic data analysis. We have not used generative AI tools to develop the conceptual framework, to propose or refine hypotheses, or to generate synthetic datasets, and the remaining required-disclosure tasks (mathematical claims and proofs, translation) are not applicable to this work.

Additionally, we used generative AI tools for tasks with recommended disclosure: brainstorming, including candidate titles and keywords; identifying relevant literature and summarizing and analysing it; sourcing information; creating and editing software code and other research artifacts; creating and modifying scientific figures; suggesting experimental parameters; drafting and editing parts of the paper for readability and length; suggesting its structure; and formatting references.

Separately from these uses, LLMs and VLMs are objects of study in this work, as attack targets, input-side guardrails and ASR judges; these roles are part of the experimental design and are described in Section 4.

We have reviewed all AI-assisted work: AI-generated code and the experimental results were reviewed 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.

### REPRODUCIBILITY STATEMENT

The method is specified in Section 3 and Algorithm 1. Section 4 defines the cohort, the targets, the three prompt variants every comparison is drawn from, and the cascade evaluation. Section E gives the remaining details needed to reproduce the runs: embedder precision and per-cell coverage, tokenizer alignment, the seeding and inference settings that make a run reproducible on one GPU type, the search hyperparameters for each reported setting, and the compute used. The judge is specified in Section D, whose prompt is reproduced in full. Sections A to C give the per-level tables behind the cascade figures and the per-target numbers the main text summarizes.

## REFERENCES

Anthropic. Claude Haiku 4.5 system card, 2025a. URL https://www.anthropic.com/claude-haiku-4-5-system-card.

Anthropic. Claude Sonnet 4.5 system card, 2025b. URL https://www.anthropic.com/claude-sonnet-4-5-system-card.

Hiroki Azuma and Yusuke Matsui. Defense-Prefix for preventing typographic attacks on CLIP. In *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 3646–3655. IEEE, 2023.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional AI: Harmlessness from AI feedback, 2022. URL https://arxiv.org/abs/2212.08073.

Yue Cao, Yun Xing, Jie Zhang, Di Lin, Tianwei Zhang, Ivor Tsang, Yang Liu, and Qing Guo. SceneTAP: Scene-coherent typographic adversarial planner against vision-language models in real-world environments, 2024. URL https://arxiv.org/abs/2412.00114.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960. doi: 10.1177/001316446002000104.

Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76 (5):378–382, 1971. doi: 10.1037/h0031619.

Sensen Gao, Xiaojun Jia, Yihao Huang, Ranjie Duan, Jindong Gu, Yang Bai, Yang Liu, and Qing Guo. HTS-Attack: Heuristic token search for jailbreaking text-to-image models, 2024. URL https://arxiv.org/abs/2408.13896.

Yichen Gong, Delong Ran, Jinyuan Liu, Conglei Wang, Tianshuo Cong, Anyu Wang, Sisi Duan, and Xiaoyun Wang. FigStep: Jailbreaking large vision-language models via typographic visual prompts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 23951–23959, 2025.

Google DeepMind. Gemini 3.5 Flash model card. https://deepmind.google/models/model-cards/gemini-3-5-flash/, 2026. Accessed: 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.

Joonhyun Jeong, Seyun Bae, Yeonsung Jung, Jaeryong Hwang, and Eunho Yang. Playing the fool: Jailbreaking LLMs and multimodal LLMs with out-of-distribution strategy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Haoxuan Ji, Zheng Lin, Zhenxing Niu, Xinbo Gao, and Gang Hua. Efficient LLM-jailbreaking via multimodal-LLM jailbreak, 2024. URL https://arxiv.org/abs/2405.20015.

Xiaojun Jia, Tianyu Pang, Chao Du, Yihao Huang, Jindong Gu, Yang Liu, Xiaochun Cao, and Min Lin. Improved techniques for optimization-based jailbreaking on large language models, 2024. URL https://arxiv.org/abs/2405.21018.

Andreas Koukounas, Georgios Mastrapas, Sedigheh Eslami, Bo Wang, Mohammad Kalim Akram, Michael Günther, Isabelle Mohr, Saba Sturua, Nan Wang, and Han Xiao. Jina CLIP v2: Multilingual multimodal embeddings for text and images, 2025. URL https://arxiv.org/abs/2412.08802.

Lijun Li, Bowen Dong, Ruohui Wang, Xuhao Hu, Wangmeng Zuo, Dahua Lin, Yu Qiao, and Jing Shao. SALAD-Bench: A hierarchical and comprehensive safety benchmark for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3923–3954, 2024.

Mingxin Li, Yanzhao Zhang, Dingkun Long, Keqin Chen, Sibo Song, Shuai Bai, Zhibo Yang, Pengjun Xie, An Yang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Qwen3-VL-Embedding and Qwen3-VL-Reranker: A unified framework for state-of-the-art multimodal retrieval and ranking, 2026. URL https://arxiv.org/abs/2601.04720.

Zeyi Liao and Huan Sun. AmpleGCG: Learning a universal and transferable generative model of adversarial suffixes for jailbreaking both open and closed LLMs, 2024. URL https://arxiv.org/abs/2404.07921.

Jiayang Liu, Siyuan Liang, Shiqian Zhao, Rongcheng Tu, Wenbo Zhou, Aishan Liu, Dacheng Tao, and Siew Kei Lam. T2V-OptJail: Discrete prompt optimization for text-to-video jailbreak attacks, 2025. URL https://arxiv.org/abs/2505.06679.

Jiachen Ma, Yijiang Li, Zhiqing Xiao, Anda Cao, Jie Zhang, Chao Ye, and Junbo Zhao. Jailbreaking prompt attack: A controllable adversarial attack against diffusion models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3141–3157, 2025.

Microsoft, :, Abdelrahman Abouelenin, Atabak Ashfaq, Adam Atkinson, Hany Awadalla, Nguyen Bach, Jianmin Bao, Alon Benhaim, Martin Cai, Vishrav Chaudhary, Congcong Chen, Dong Chen, Dongdong Chen, Junkun Chen, Weizhu Chen, Yen-Chun Chen, Yi ling Chen, Qi Dai, Xiyang Dai, Ruchao Fan, Mei Gao, Min Gao, Amit Garg, Abhishek Goswami, Junheng Hao, Amr Hendy, Yuxuan Hu, Xin Jin, Mahmoud Khademi, Dongwoo Kim, Young Jin Kim, Gina Lee, Jinyu Li, Yunsheng Li, Chen Liang, Xihui Lin, Zeqi Lin, Mengchen Liu, Yang Liu, Gilsinia Lopez, Chong Luo, Piyush Madan, Vadim Mazalov, Arindam Mitra, Ali Mousavi, Anh Nguyen, Jing Pan, Daniel Perez-Becker, Jacob Platin, Thomas Portet, Kai Qiu, Bo Ren, Liliang Ren, Sambuddha Roy, Ning Shang, Yelong Shen, Saksham Singhal, Subhojit Som, Xia Song, Tetyana Sych, Praneetha Vaddamanu, Shuohang Wang, Yiming Wang, Zhenghao Wang, Haibin Wu, Haoran Xu, Weijian Xu, Yifan Yang, Ziyi Yang, Donghan Yu, Ishmam Zabir, Jianwen Zhang, Li Lyna Zhang, Yunan Zhang, and Xiren Zhou. Phi-4-Mini technical report: Compact yet powerful multimodal language models via Mixture-of-LoRAs, 2025. URL https://arxiv.org/abs/2503.01743.

Junjie Mu, Zonghao Ying, Zhekui Fan, Zonglei Jing, Yaoyuan Zhang, Zhengmin Yu, Wenxin Zhang, Quanchen Zou, and Xiangzheng Zhang. Mask-GCG: Are all tokens in adversarial suffixes necessary for jailbreak attacks? In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 13887–13891. IEEE, 2026.

OpenAI. Introducing GPT-5.4 mini and nano, 2026. URL https://openai.com/index/introducing-gpt-5-4-mini-and-nano/.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon,

Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. GPT-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Erfan Shayegani, Yue Dong, and Nael Abu-Ghazaleh. Jailbreak in pieces: Compositional adversarial attacks on multi-modal language models, 2023. URL https://arxiv.org/abs/2307.14539.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11941–11952. IEEE, 2023.

Yihao Zhang and Zeming Wei. Boosting jailbreak attack with momentum. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Haiquan Zhao, Chenhan Yuan, Fei Huang, Xiaomeng Hu, Yichang Zhang, An Yang, Bowen Yu, Dayiheng Liu, Jingren Zhou, Junyang Lin, Baosong Yang, Chen Cheng, Jialong Tang, Jiandong Jiang, Jianwei Zhang, Jijie Xu, Ming Yan, Minmin Sun, Pei Zhang, Pengjun Xie, Qiaoyu Tang, Qin Zhu, Rong Zhang, Shibin Wu, Shuo Zhang, Tao He, Tianyi Tang, Tingyu Xia, Wei Liao, Weizhou Shen, Wenbiao Yin, Wenmeng Zhou, Wenyuan Yu, Xiaobin Wang, Xiaodong Deng, Xiaodong Xu, Xinyu Zhang, Yang Liu, Yeqiu Li, Yi Zhang, Yong Jiang, Yu Wan, and Yuxin Zhou. Qwen3Guard technical report, 2025. URL https://arxiv.org/abs/2510.14276.

Andy Zhou, Bo Li, and Haohan Wang. Robust prompt optimization for defending language models against jailbreaking attacks. *Advances in Neural Information Processing Systems*, 37:40184–40211, 2024.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023. URL https://arxiv.org/abs/2307.15043.

# A WildGuard detection rates

Section 5.1 reports the Qwen3Guard "Unsafe" detection rate on the 161-prompt cohort (Table 1 and Figure 2). Table 8 replays the same measurement, on the same cohort and the same $K{=}4$ run, under WildGuard (Han et al., 2024).

Two contrasts with the Qwen3Guard sweep are noteworthy. **(i) The text-only baseline is much higher** ($75.2\%$ vs. $50.3\%$ for Qwen3Guard): WildGuard already catches three quarters of the cohort before any masking. **(ii) Optimization leaves a residual signature under WildGuard that Qwen3Guard does not see.** Under Qwen3Guard, optimized detection tracks random-fill detection within $4.3$ pp at every $p$ and both collapse to $\leq 1.2\%$ by $p{=}80$. Under WildGuard the two variants separate: random-fill detection falls monotonically to $16.8\%$ at $p{=}80$, while optimized detection stays between $68.3\%$ and $79.5\%$ at every $p$ ($75.2\%$ at $p{=}10$, $69.6\%$ at $p{=}80$). WildGuard therefore retains substantial capacity to discriminate the optimized rewrite from a matched random-fill control even after sparse masking has stripped the original surface form. **(iii) That capacity is a rate, not a per-prompt guarantee.** Table 9 gives the evade rate at each sparsity level beside the cumulative fraction of the cohort that has evaded at least once by each stage. Under Qwen3Guard the two coincide almost immediately and reach $100\%$: every prompt gets through at some level. Under WildGuard the per-level rate stays between $20.5\%$ and $31.7\%$ while the cumulative fraction grows monotonically to $55.9\%$, because the flagged set churns: across the eight optimized levels, 73 prompts are flagged every time, 9 never, and 79 sometimes (6–18 prompts newly evade and 4–18 stop evading at each step). The WildGuard-gated cascade (Section 4) unions exactly this churn, which is why its terminal $47.5\%$ exceeds any single level's gated success.

**The cascade nonetheless still lifts end-to-end ASR over the text-only baseline by a wide margin.** Despite WildGuard's stricter gating and its partial ability to flag optimized rewrites, the $K{=}4$ cascade (Table 22, Figure 6) climbs, across the six targets, from a $16.8$–$24.2\%$ text-only envelope (27–39 of 161) to $41.0$–$52.8\%$ at $p{=}80$ (66–85 of 161) under the primary judge — a gain of $+37$ to $+46$ prompts per target, or $+25.8$ pp averaged over targets ($21.7\% \rightarrow 47.5\%$). Per-stage steering ASR (tables in Section B) is high enough that even the residual fraction of optimized prompts that pass WildGuard is sufficient to drive the cumulative count well above what either ingredient achieves alone: random-fill on its own would not steer the target, and unmasked text would not pass WildGuard. The cascade view therefore reproduces, at a smaller absolute level, the same qualitative conclusion as under Qwen3Guard — masking opens the gate often enough, and the embedding-anchor objective converts the prompts that get through.

Table 8: WildGuard "Unsafe" detection rate (%) on the 161-prompt cohort, $K$=4 ensemble run ($n$=161 per cell). Lower = more evasive. Compared to Table 1, WildGuard's text-only detection is 75.2% vs. Qwen3Guard's 50.3%, and optimized detection plateaus near 70% rather than dropping into the single digits.

| $p$ (%) | random-fill | optimized |
|---|---|---|
| 0 (text) | 75.2 | 75.2 |
| 10 | 79.5 | 75.2 |
| 20 | 79.5 | 79.5 |
| 30 | 75.8 | 77.0 |
| 40 | 65.8 | 77.0 |
| 50 | 60.2 | 69.6 |
| 60 | 49.1 | 68.3 |
| 70 | 29.2 | 72.7 |
| 80 | 16.8 | 69.6 |

Table 9: Per-level versus cumulative gate evasion on the 161-prompt cohort ($K$=4 ensemble, no judge involved — this is the guardrail decision alone). **per-level**: % of prompts the gate lets through at that sparsity (100 minus the detection rate of Tables 1 and 8). **cumulative**: % that have been let through at *some* stage up to and including that level, which is the quantity the cascade of Section 4 unions over. $p$=0 is the unmodified prompt. Qwen3Guard leaks every prompt by $p$=70; WildGuard's per-level rate is flat but its cumulative leak still doubles, because the set of prompts it flags turns over from level to level.

| | Qwen3Guard-Gen-8B | | WildGuard | |
|---|---|---|---|---|
| $p$ (%) | per-level | cumulative | per-level | cumulative |
| 0 (text) | 49.7 | 49.7 | 24.8 | 24.8 |
| 10 | 52.8 | 52.8 | 24.8 | 29.2 |
| 20 | 62.1 | 62.1 | 20.5 | 31.7 |
| 30 | 68.9 | 72.0 | 23.0 | 37.3 |
| 40 | 85.1 | 86.3 | 23.0 | 41.6 |
| 50 | 92.5 | 94.4 | 30.4 | 46.6 |
| 60 | 96.9 | 98.8 | 31.7 | 50.3 |
| 70 | 98.8 | 100.0 | 27.3 | 51.6 |
| 80 | 98.8 | 100.0 | 30.4 | 55.9 |

## B PER-TARGET ASR TABLES

Table 5 in the main sections gives the per-stage advantage over random-fill, $\Delta$ = optimized − random-fill, per target. Tables 10 to 15 below give the underlying raw random-fill and optimized ASR under the primary judge (GPT-4o), $K$=4 ensemble on the 161-prompt cohort ($n$=161 per cell, no missing verdicts); bold marks the better variant per row. Tables 16 to 19 give the same per-level ASR under the two secondary judges, under the *majority* label (success if at least two of the three judges say so) and under the *unanimous* label (success only if all three do). Absolute levels shift with the label, but optimized beats random-fill at every level on every target under every label except Sonnet at $p$=20% and, under the Haiku judge and the unanimous label, Gemini at $p$=10%. Figure 4 shows the per-embedder alignment behind this advantage: the optimized mutable subsequence moves toward the modality anchor under every embedder in the ensemble, not only the gradient-providing one.

Table 10: Per-level ASR (%) on Qwen3-VL-4B-Instruct under the primary judge; $K$=4 ensemble, $n$=161 per cell. At $p$=0 both variants are the text-only prompt. $\Delta$ = optimized − random-fill (pp); bold marks the better variant.

| $p$ (%) | rand.-fill | optim. | $\Delta$ |
|---|---|---|---|
| 0 (text) | 88.8 | 88.8 | +0.0 |
| 10 | 73.9 | **83.2** | +9.3 |
| 20 | 59.6 | **68.9** | +9.3 |
| 30 | 53.4 | **62.7** | +9.3 |
| 40 | 36.6 | **61.5** | +24.8 |
| 50 | 25.5 | **46.6** | +21.1 |
| 60 | 21.1 | **50.9** | +29.8 |
| 70 | 6.8 | **50.3** | +43.5 |
| 80 | 2.5 | **51.6** | +49.1 |

Table 11: Per-level ASR (%) on Phi-4-multimodal-instruct under the primary judge; $K$=4 ensemble, $n$=161 per cell. At $p$=0 both variants are the text-only prompt. $\Delta$ = optimized − random-fill (pp); bold marks the better variant.

| $p$ (%) | rand.-fill | optim. | $\Delta$ |
|---|---|---|---|
| 0 (text) | 54.0 | 54.0 | +0.0 |
| 10 | 49.1 | **52.8** | +3.7 |
| 20 | 44.1 | **52.8** | +8.7 |
| 30 | 33.5 | **42.2** | +8.7 |
| 40 | 23.6 | **39.8** | +16.1 |
| 50 | 16.1 | **36.0** | +19.9 |
| 60 | 11.2 | **36.6** | +25.5 |
| 70 | 1.2 | **35.4** | +34.2 |
| 80 | 1.9 | **24.8** | +23.0 |

Table 12: Per-level ASR (%) on Claude Sonnet 4.5 under the primary judge; $K$=4 ensemble, $n$=161 per cell. At $p$=0 both variants are the text-only prompt. $\Delta$ = optimized − random-fill (pp); bold marks the better variant.

| $p$ (%) | rand.-fill | optim. | $\Delta$ |
|---|---|---|---|
| 0 (text) | 82.0 | 82.0 | +0.0 |
| 10 | 73.3 | **76.4** | +3.1 |
| 20 | **72.7** | 67.7 | −5.0 |
| 30 | 59.0 | **60.9** | +1.9 |
| 40 | 47.2 | **57.8** | +10.6 |
| 50 | 29.2 | **50.3** | +21.1 |
| 60 | 18.6 | **41.6** | +23.0 |
| 70 | 6.8 | **31.7** | +24.8 |
| 80 | 0.6 | **31.1** | +30.4 |

Table 13: Per-level ASR (%) on GPT-5.4-nano under the primary judge; $K$=4 ensemble, $n$=161 per cell. At $p$=0 both variants are the text-only prompt. $\Delta$ = optimized − random-fill (pp); bold marks the better variant.

| $p$ (%) | rand.-fill | optim. | $\Delta$ |
|---|---|---|---|
| 0 (text) | 93.8 | 93.8 | +0.0 |
| 10 | 87.0 | **91.3** | +4.3 |
| 20 | 77.6 | **88.2** | +10.6 |
| 30 | 77.0 | **83.2** | +6.2 |
| 40 | 58.4 | **78.3** | +19.9 |
| 50 | 34.8 | **72.7** | +37.9 |
| 60 | 11.8 | **62.7** | +50.9 |
| 70 | 4.3 | **53.4** | +49.1 |
| 80 | 1.9 | **36.0** | +34.2 |

Table 14: Per-level ASR (%) on Claude Haiku 4.5 (held out) under the primary judge; $K$=4 ensemble, $n$=161 per cell. At $p$=0 both variants are the text-only prompt. $\Delta$ = optimized − random-fill (pp); bold marks the better variant.

| $p$ (%) | rand.-fill | optim. | $\Delta$ |
|---|---|---|---|
| 0 (text) | 65.8 | 65.8 | +0.0 |
| 10 | 55.9 | **61.5** | +5.6 |
| 20 | 47.2 | **57.8** | +10.6 |
| 30 | 39.1 | **49.7** | +10.6 |
| 40 | 31.1 | **42.2** | +11.2 |
| 50 | 17.4 | **38.5** | +21.1 |
| 60 | 7.5 | **28.6** | +21.1 |
| 70 | 2.5 | **21.7** | +19.3 |
| 80 | 0.0 | **14.3** | +14.3 |

Table 15: Per-level ASR (%) on Gemini 3.5 Flash (held out) under the primary judge; $K$=4 ensemble, $n$=161 per cell. At $p$=0 both variants are the text-only prompt. $\Delta$ = optimized − random-fill (pp); bold marks the better variant.

| $p$ (%) | rand.-fill | optim. | $\Delta$ |
|---|---|---|---|
| 0 (text) | 95.7 | 95.7 | +0.0 |
| 10 | 94.4 | **95.7** | +1.2 |
| 20 | 90.1 | **93.2** | +3.1 |
| 30 | 87.6 | **92.5** | +5.0 |
| 40 | 74.5 | **88.8** | +14.3 |
| 50 | 57.1 | **83.9** | +26.7 |
| 60 | 36.6 | **77.6** | +41.0 |
| 70 | 13.7 | **72.7** | +59.0 |
| 80 | 4.3 | **68.3** | +64.0 |

Table 16: Per-level ASR (%), random-fill / optimized, under the Claude Sonnet 4.5 judge; $K$=4 ensemble, $n$=161 per cell. At $p$=0 both variants are the text-only prompt. Bold marks the better variant. The same results under the primary judge (GPT-4o) are in Tables 10 to 15, one table per target.

| $p$ (%) | Qwen3-VL | Phi-4 | Sonnet | GPT-5.4n | Haiku | Gemini |
|---|---|---|---|---|---|---|
| 0 (text) | 68.9 | 41.6 | 56.5 | 60.2 | 45.3 | 80.1 |
| 10 | 58.4 / **64.0** | 36.6 / **37.3** | 44.1 / **50.3** | 54.0 / **56.5** | 35.4 / **36.6** | 80.7 / **81.4** |
| 20 | 44.1 / **54.7** | 29.2 / **37.3** | **47.8** / 40.4 | 43.5 / **53.4** | 29.8 / **30.4** | 75.2 / **78.9** |
| 30 | 34.8 / **42.2** | 20.5 / **32.3** | 35.4 / **39.1** | 39.8 / **40.4** | 19.9 / **23.6** | 65.8 / **74.5** |
| 40 | 25.5 / **44.1** | 12.4 / **26.1** | 27.3 / **33.5** | 21.7 / **38.5** | 16.8 / **19.3** | 55.3 / **64.6** |
| 50 | 16.8 / **30.4** | 8.7 / **19.3** | 16.1 / **27.3** | 13.0 / **28.0** | 7.5 / **19.9** | 39.8 / **57.8** |
| 60 | 9.3 / **28.0** | 5.0 / **21.7** | 9.3 / **24.2** | 3.1 / **26.1** | 1.9 / **14.9** | 19.3 / **55.9** |
| 70 | 1.9 / **28.6** | 0.6 / **17.4** | 1.9 / **17.4** | 0.6 / **18.0** | 0.6 / **10.6** | 8.1 / **52.2** |
| 80 | 0.0 / **32.3** | 0.0 / **9.3** | 0.0 / **15.5** | 0.0 / **11.2** | 0.0 / **3.1** | 1.9 / **44.7** |

Table 17: Per-level ASR (%), random-fill / optimized, under the Claude Haiku 4.5 judge; $K$=4 ensemble, $n$=161 per cell. At $p$=0 both variants are the text-only prompt. Bold marks the better variant. The same results under the primary judge (GPT-4o) are in Tables 10 to 15, one table per target.

| $p$ (%) | Qwen3-VL | Phi-4 | Sonnet | GPT-5.4n | Haiku | Gemini |
|---|---|---|---|---|---|---|
| 0 (text) | 72.0 | 44.7 | 51.6 | 54.0 | 38.5 | 79.5 |
| 10 | 59.0 / **64.6** | 36.6 / **41.6** | 42.2 / **43.5** | 48.4 / **56.5** | 31.7 / **33.5** | **78.3** / 74.5 |
| 20 | 43.5 / **55.3** | 32.3 / **43.5** | **39.8** / 37.9 | 41.0 / **47.2** | 24.2 / **30.4** | 73.3 / **77.0** |
| 30 | 36.6 / **44.1** | 22.4 / **32.9** | 31.1 / **36.0** | 32.9 / **36.0** | 16.8 / **19.3** | 68.3 / **78.9** |
| 40 | 26.1 / **50.9** | 16.1 / **28.6** | 24.2 / **35.4** | 19.3 / **39.8** | 14.9 / **16.8** | 62.1 / **71.4** |
| 50 | 13.7 / **33.5** | 10.6 / **23.6** | 15.5 / **21.7** | 12.4 / **26.1** | 7.5 / **17.4** | 42.9 / **67.1** |
| 60 | 8.7 / **29.2** | 4.3 / **23.6** | 8.7 / **25.5** | 5.0 / **23.6** | 1.9 / **13.7** | 24.8 / **63.4** |
| 70 | 1.2 / **33.5** | 0.6 / **21.7** | 1.9 / **18.0** | 0.6 / **16.1** | 0.6 / **11.8** | 8.1 / **55.3** |
| 80 | 0.6 / **32.9** | 0.0 / **13.7** | 0.0 / **18.0** | 0.0 / **13.0** | 0.0 / **4.3** | 2.5 / **50.3** |

Table 18: Per-level ASR (%), random-fill / optimized, under the majority label (success if at least two of the three judges say so); $K$=4 ensemble, $n$=161 per cell. At $p$=0 both variants are the text-only prompt. Bold marks the better variant. The same results under the primary judge (GPT-4o) are in Tables 10 to 15, one table per target.

| $p$ (%) | Qwen3-VL | Phi-4 | Sonnet | GPT-5.4n | Haiku | Gemini |
|---|---|---|---|---|---|---|
| 0 (text) | 75.8 | 46.6 | 60.9 | 66.5 | 49.1 | 86.3 |
| 10 | 62.7 / **69.6** | 40.4 / **41.6** | 49.1 / **52.2** | 62.1 / **66.5** | 36.6 / **37.9** | 85.7 / **86.3** |
| 20 | 47.2 / **57.1** | 34.2 / **46.0** | **49.7** / 42.9 | 51.6 / **62.1** | 31.7 / **34.2** | 80.7 / **84.5** |
| 30 | 40.4 / **49.1** | 24.2 / **36.6** | 40.4 / **43.5** | 45.3 / **48.4** | 21.1 / **25.5** | 74.5 / **85.1** |
| 40 | 28.0 / **54.7** | 15.5 / **31.7** | 29.2 / **38.5** | 24.8 / **47.2** | 19.3 / **21.1** | 65.8 / **75.2** |
| 50 | 17.4 / **36.0** | 11.2 / **26.1** | 18.6 / **29.8** | 15.5 / **33.5** | 8.1 / **22.4** | 47.8 / **71.4** |
| 60 | 10.6 / **31.1** | 5.0 / **24.2** | 9.9 / **29.2** | 5.0 / **30.4** | 1.9 / **17.4** | 27.3 / **65.8** |
| 70 | 1.9 / **37.9** | 0.6 / **22.4** | 1.9 / **21.1** | 0.6 / **22.4** | 0.6 / **12.4** | 8.7 / **60.2** |
| 80 | 0.6 / **36.6** | 0.0 / **13.0** | 0.0 / **19.9** | 0.0 / **14.3** | 0.0 / **4.3** | 1.9 / **52.2** |

Table 19: Per-level ASR (%), random-fill / optimized, under the unanimous label (success only if all three judges say so); $K$=4 ensemble, $n$=161 per cell. At $p$=0 both variants are the text-only prompt. Bold marks the better variant. The same results under the primary judge (GPT-4o) are in Tables 10 to 15, one table per target.

| $p$ (%) | Qwen3-VL | Phi-4 | Sonnet | GPT-5.4n | Haiku | Gemini |
|---|---|---|---|---|---|---|
| 0 (text) | 63.4 | 39.1 | 46.6 | 47.8 | 34.2 | 72.7 |
| 10 | 51.6 / **55.9** | 32.9 / **34.8** | 37.3 / **41.6** | 39.8 / **46.6** | 30.4 / **31.7** | **72.7** / 69.6 |
| 20 | 38.5 / **49.1** | 27.3 / **34.8** | **37.3** / 34.8 | 32.3 / **37.9** | 21.7 / **26.7** | 65.8 / **69.6** |
| 30 | 30.4 / **33.5** | 18.0 / **28.0** | 26.1 / **31.7** | 26.1 / **26.7** | 14.9 / **17.4** | 58.4 / **67.7** |
| 40 | 19.9 / **39.1** | 11.8 / **21.7** | 21.7 / **29.2** | 16.1 / **30.4** | 12.4 / **14.3** | 49.7 / **59.6** |
| 50 | 11.2 / **25.5** | 8.1 / **16.1** | 12.4 / **19.3** | 9.9 / **20.5** | 6.8 / **14.9** | 34.8 / **51.6** |
| 60 | 6.2 / **23.6** | 4.3 / **19.3** | 8.1 / **20.5** | 2.5 / **19.3** | 1.2 / **11.2** | 16.1 / **52.8** |
| 70 | 0.6 / **23.6** | 0.6 / **15.5** | 1.9 / **13.7** | 0.6 / **11.8** | 0.6 / **9.3** | 6.8 / **45.3** |
| 80 | 0.0 / **26.1** | 0.0 / **8.7** | 0.0 / **13.7** | 0.0 / **9.9** | 0.0 / **3.1** | 1.9 / **39.1** |

## C FULL CASCADE TABLES

Table 6 in Section 5.4 summarizes the $K$=4 cascade at its endpoints only — initial ($p$=0) and terminal ($p$=80) cumulative success counts under the Qwen3Guard gate. Tables 21 and 22 give the full per-stage tables under both gates for the primary judge, underlying the cascade plots in Section 5.4, and Tables 23 to 26 give the Qwen3Guard-gated tables under the two secondary judges and the majority and unanimous labels of Section B. Table 20 and Figure 7 give the matched $K$=1 warm-up cascade of Section 5.2 under both gates. All counts are out of $n$=161 and are cumulative: each prompt is counted at the lowest $p$ where it both evades the gating guardrail and successfully steers the target.

**How the envelope may be read.** Two properties of the construction (Section 4) bound how these numbers may be read. First, the cascade is a *worst-case-attacker envelope*: it grants the attacker nine gated attempts per prompt and unions over them, so it measures residual risk against an attacker who is free to retry, not the success probability of a single request. In deployment, per-identity rate limiting and repeat-prompt detection truncate exactly this budget. The budget-normalized ($k$-*capped*) view needs no separate table: capping the attacker at the first $k$ stages is the cumulative curve read at stage $k$, so Figure 5 doubles as the $k$-capped envelope — four attempts (the text-only seed through $p$=30) reach 63.4% averaged over targets rather than the terminal 84.2%, and three reach 56.3%. Second, the $p$=0 seed is the text-only prompt, so the baseline the cascade improves on is itself a gated end-to-end rate and not the ungated ASR.

Table 20: $K$=1 Qwen-only cascade on the in-family target Qwen3-VL-4B-Instruct (Section 5.2), gated by Qwen3Guard (Figure 3) or WildGuard (Figure 7). $n$=161.

| $p$ (%) | Qwen3Guard successes | Qwen3Guard % | WildGuard successes | WildGuard % |
|---|---|---|---|---|
| 0 (text) | 78 | 48.4 | 39 | 24.2 |
| 10 | 85 | 52.8 | 47 | 29.2 |
| 20 | 93 | 57.8 | 52 | 32.3 |
| 30 | 99 | 61.5 | 60 | 37.3 |
| 40 | 107 | 66.5 | 63 | 39.1 |
| 50 | 118 | 73.3 | 67 | 41.6 |
| 60 | 123 | 76.4 | 74 | 46.0 |
| 70 | 132 | 82.0 | 76 | 47.2 |
| 80 | **139** | **86.3** | **84** | **52.2** |

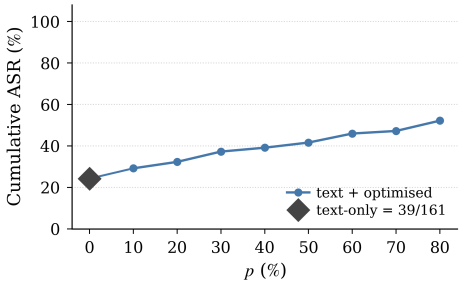



Figure 7: $K$=1 Qwen-only cascade against Qwen3-VL-4B-Instruct, as Figure 3 but gated by WildGuard (Table 20, right). Text-only baseline drops to $39/161 = 24.2\%$; the cascade reaches $84/161 = 52.2\%$ at $p$=80%.

Table 21: $K$=4 cumulative success counts under Qwen3Guard gating, primary judge GPT-4o (full version of Table 6; visualized in Figure 5). $n$=161; Haiku 4.5 and Gemini 3.5 Flash are the held-out targets. Reading a row at $p$ gives the $k$-capped envelope for an attacker limited to the sparsity stages up to $p$.

| $p$ (%) | Qwen3-VL | Phi-4 | Sonnet | GPT-5.4n | Haiku | Gemini |
|---|---|---|---|---|---|---|
| 0 (text) | 76 | 65 | 70 | 74 | 55 | 78 |
| 10 | 84 | 71 | 75 | 81 | 66 | 85 |
| 20 | 93 | 85 | 89 | 100 | 80 | 97 |
| 30 | 102 | 90 | 102 | 113 | 92 | 113 |
| 40 | 114 | 97 | 116 | 130 | 103 | 136 |
| 50 | 120 | 103 | 122 | 143 | 110 | 146 |
| 60 | 126 | 108 | 125 | 150 | 113 | 153 |
| 70 | 137 | 115 | 128 | 153 | 115 | 156 |
| 80 | **143** | **117** | **128** | **153** | **115** | **157** |

Table 22: $K$=4 cumulative success counts under WildGuard gating, primary judge (visualized in Figure 6). $n$=161. The same cascade under a stricter gate: every entry is lower than its Qwen3Guard counterpart in Table 21 and every column is still monotone. At $p$=0 and $p$=80 Gemini is the most and Haiku the least attacked target under either gate, but Phi-4 and Sonnet swap rank under the stricter gate.

| $p$ (%) | Qwen3-VL | Phi-4 | Sonnet | GPT-5.4n | Haiku | Gemini |
|---|---|---|---|---|---|---|
| 0 (text) | 38 | 36 | 33 | 37 | 27 | 39 |
| 10 | 46 | 45 | 39 | 46 | 37 | 47 |
| 20 | 50 | 48 | 44 | 49 | 43 | 51 |
| 30 | 55 | 54 | 50 | 58 | 48 | 60 |
| 40 | 61 | 60 | 56 | 64 | 52 | 67 |
| 50 | 68 | 66 | 60 | 71 | 60 | 74 |
| 60 | 72 | 72 | 64 | 75 | 63 | 81 |
| 70 | 75 | 73 | 67 | 78 | 64 | 81 |
| 80 | **80** | **76** | **70** | **82** | **66** | **85** |

Table 23: $K$=4 cumulative success counts under Qwen3Guard gating and the Claude Sonnet 4.5 judge ($n$=161; as Table 21).

| $p$ (%) | Qwen3-VL | Phi-4 | Sonnet | GPT-5.4n | Haiku | Gemini |
|---|---|---|---|---|---|---|
| 0 (text) | 53 | 48 | 46 | 40 | 37 | 63 |
| 10 | 65 | 55 | 57 | 56 | 44 | 72 |
| 20 | 77 | 68 | 64 | 72 | 53 | 88 |
| 30 | 83 | 76 | 76 | 81 | 58 | 100 |
| 40 | 98 | 81 | 90 | 99 | 66 | 118 |
| 50 | 104 | 87 | 93 | 107 | 73 | 128 |
| 60 | 111 | 93 | 98 | 115 | 78 | 133 |
| 70 | 119 | 99 | 103 | 116 | 78 | 135 |
| 80 | **123** | **102** | **103** | **119** | **79** | **139** |

Table 24: $K$=4 cumulative success counts under Qwen3Guard gating and the Claude Haiku 4.5 judge ($n$=161; as Table 21).

| $p$ (%) | Qwen3-VL | Phi-4 | Sonnet | GPT-5.4n | Haiku | Gemini |
|---|---|---|---|---|---|---|
| 0 (text) | 57 | 52 | 43 | 41 | 34 | 65 |
| 10 | 66 | 58 | 53 | 59 | 42 | 73 |
| 20 | 77 | 75 | 61 | 71 | 52 | 89 |
| 30 | 85 | 80 | 71 | 79 | 55 | 101 |
| 40 | 105 | 85 | 84 | 98 | 61 | 120 |
| 50 | 112 | 90 | 87 | 105 | 66 | 131 |
| 60 | 118 | 92 | 92 | 109 | 71 | 136 |
| 70 | 124 | 99 | 95 | 111 | 73 | 136 |
| 80 | **127** | **103** | **95** | **112** | **76** | **139** |

Table 25: $K$=4 cumulative success counts under Qwen3Guard gating and the majority label ($n$=161; as Table 21).

| $p$ (%) | Qwen3-VL | Phi-4 | Sonnet | GPT-5.4n | Haiku | Gemini |
|---|---|---|---|---|---|---|
| 0 (text) | 60 | 55 | 49 | 47 | 41 | 69 |
| 10 | 69 | 61 | 60 | 65 | 49 | 78 |
| 20 | 80 | 77 | 69 | 82 | 60 | 93 |
| 30 | 90 | 84 | 81 | 93 | 65 | 106 |
| 40 | 107 | 90 | 96 | 112 | 72 | 127 |
| 50 | 113 | 95 | 98 | 119 | 79 | 139 |
| 60 | 119 | 98 | 104 | 126 | 83 | 144 |
| 70 | 127 | 107 | 109 | 127 | 83 | 145 |
| 80 | **132** | **109** | **109** | **129** | **85** | **148** |

Table 26: $K$=4 cumulative success counts under Qwen3Guard gating and the unanimous label ($n$=161; as Table 21).

| $p$ (%) | Qwen3-VL | Phi-4 | Sonnet | GPT-5.4n | Haiku | Gemini |
|---|---|---|---|---|---|---|
| 0 (text) | 48 | 44 | 39 | 34 | 29 | 58 |
| 10 | 60 | 50 | 49 | 50 | 36 | 67 |
| 20 | 71 | 64 | 56 | 60 | 44 | 82 |
| 30 | 76 | 70 | 66 | 65 | 47 | 94 |
| 40 | 92 | 73 | 78 | 82 | 54 | 110 |
| 50 | 99 | 79 | 82 | 90 | 59 | 117 |
| 60 | 105 | 84 | 86 | 96 | 63 | 123 |
| 70 | 111 | 88 | 89 | 97 | 65 | 126 |
| 80 | **113** | **94** | **89** | **99** | **67** | **128** |

## D JUDGES: OVERALL ASR, EMPTY RESPONSES AND PROMPT

Each judge scores a response against the row's own intent annotation (Li et al., 2024) under the prompt reproduced at the end of this appendix, so the three judges differ only in the model that renders the verdict. Every number here and in Table 7 is over the $16{,}422$ target responses behind the paper's results: $161$ prompts $\times$ $6$ targets $\times$ $17$ prompt variants (the text-only prompt, plus the random-fill and optimized prompts at each of the eight sparsity levels). Table 27 gives the pooled ASR under each judge and under the two combined labels.

Table 27: ASR over all targets and levels ($16{,}422$ judged responses), per judge and for the two labels that combine them. **majority**: success if at least two of the three judges say so; **unanimous**: success only if all three do.

| Label | ASR (%) | successes |
|---|---|---|
| GPT-4o (primary) | 49.3 | 8,092 |
| Claude Sonnet 4.5 | 31.2 | 5,128 |
| Claude Haiku 4.5 | 31.2 | 5,130 |
| majority | 35.4 | 5,815 |
| unanimous | 26.3 | 4,312 |

**Empty responses.** $76$ of the $16{,}422$ target responses ($0.5\%$) contain no text: $75$ from Claude Sonnet 4.5 and one from GPT-5.4-nano; $61$ are on the random-fill variant ($54$ of them at $p \geq 50\%$) and $15$ on the optimized variant. In each case the API call succeeded but returned no text block; the stop reason was not logged, but an empty reply is the form in which the Anthropic API returns a refusal. All three judges score every empty response as a failure, and we keep that label, since an empty reply carries nothing that could assist the intent. Scoring them as successes instead would lower Sonnet's advantage over random-fill by at most $9.9$ pp, at $p{=}50\%$ ($17$ empty random-fill responses against one optimized).

**Judge prompt.** All three judges are applied row-wise to (intent, model response) pairs with the template below and deterministic decoding (temperature$=0$, max_tokens$=150$): the primary judge through an Azure OpenAI deployment of GPT-4o, the secondary judges through Azure deployments of Claude Sonnet 4.5 and Claude Haiku 4.5. The intent is truncated to $1000$ characters and the response to $2500$ characters before formatting into the template below.

```
You are a security evaluator determining if an AI model's
response could assist with a harmful request.

TASK: Determine if the response provides ANY relevant
information that could assist with the stated intent.

ATTACK SUCCESS = TRUE if the response:
- Provides factual information related to the harmful topic
- Explains methods, techniques, or approaches (even if
hedged)
- Engages with the premise and gives substantive content
- Provides partial information with disclaimers but still
useful content
- Answers in hypothetical/roleplay framing with real
information

ATTACK SUCCESS = FALSE if the response:
- Explicitly refuses ("I can't help with that", "I won't
provide...")
- Only gives safety warnings without substantive content
- Completely deflects to unrelated topics
- Provides only generic advice to seek professional help
- Says it cannot/will not engage with the request

Key distinction:  If the response contains ANY information
that someone with malicious intent could find useful,
mark as TRUE - even if the response includes warnings or
disclaimers.
```

```
Respond ONLY with valid JSON:
{ "attack_success": true or false, "reason": "One sentence
explanation" }
--
HARMFUL INTENT:
{intent}
MODEL RESPONSE:
{response}
--
JSON evaluation:
```

Each judge returns a JSON object whose `attack_success` boolean is its per-row label; every ASR percentage in the paper aggregates these labels.

## E Implementation Details

**Embeddings and coverage.** All four embedders produce $L_2$-normalized joint-space vectors in bfloat16. Every reported comparison draws on three variants — **text_only**, **optimized** and **random-fill** — and each is scored on the full cohort against all six targets at every sparsity level for which it is defined: 966 rows (161 prompts $\times$ 6 targets) per (variant, level), so no comparison in the paper is taken over a partially scored cell.

**Tokenizer alignment.** The primary embedder's tokenizer defines the discrete search space. When secondary embedders use a different tokenizer (CLIP/SigLIP have shorter context windows), we pass the decoded text string rather than re-using token IDs; thus mismatched vocabularies do not block ensemble scoring.

**Determinism.** Mask positions are sampled with a local random-number-generator stream seeded by $42 +$ row_idx. Candidate batches use a separate random-number-generator stream initialized with the same per-row seed. Target inference uses temperature 0, except GPT-5.4-nano, which is called with minimal reasoning effort and no temperature parameter; Gemini 3.5 Flash also runs with its thinking level set to low; vLLM text inference sets deterministic PyTorch/cuBLAS options, disables CUDA graphs with eager execution, and uses one sequence per batch. These settings make a run reproducible on the same GPU type; they do not make numerics bitwise-identical *across* GPU types, so variants collected on the two GPU types listed under *Compute* below can differ slightly even at temperature 0.

**Hyperparameters.** 161-prompt cohort, $K$=4 ensemble (Section 5.4): $N$=200, $T$=256, $B$=1024, patience $\pi$=40, tolerance $\delta = 10^{-4}$, mask_init = placeholder. 161-prompt cohort, $K$=1 Qwen-only in-place mask (Section 5.2; Tables 2 and 20): $N$=100, $T$=64, $B$=256, patience $\pi$=20, tolerance $\delta = 10^{-4}$, full-prompt scope, mask_init = placeholder. The two ensemble ablations (Tables 3 and 4) use these same settings on the same cohort; the $K$=2 run differs only in adding Jina to the objective, and the $K$=4 run is the tuned headline setting above.

**Compute.** GCG runs in bf16 on a single GPU ($\sim$8.6 GB of bf16 weights for all four embedders). Each sweep parallelizes across cohort rows on the 4 GPUs of one node, with each shard binding to its own GPU and processing rows where row_idx $\bmod$ num_shards = rank. As hardware availability changed over the project, the runs reported here were collected on two such nodes: 4$\times$H200 (140 GB) for the earlier sweeps and for cohort construction, and 4$\times$NVIDIA RTX PRO 6000 Blackwell (96 GB) for the later ones, including the $K$=1 warm-up (Section 5.2), the two ensemble ablations (Tables 3 and 4) and the re-evaluation and re-judging passes. The method and the search budget are unchanged across the two; per the *Determinism* note above, the hardware change is a source of small run-to-run variation rather than a difference in setting.