CrossAnchor: Image-Anchored Text Optimization Exposes Blind Spots in Multi-line Defenses of Agentic Systems

Anish Kiran Kulkarni [1] **Ravikumar Balakrishnan** [1] **Prashanth Arun** [1]

Abstract

Input-side defenses for agentic LLM and VLM deployments are typically evaluated against discrete prompt attacks that trip guardrails or require white-box access to deployed model; multimodal jailbreaks bypass text classifiers but deliver payload as an image, limiting them to vision-enabled models. Neither family optimizes text tokens directly against a non-text embedding anchor. Our method swaps target-model log-likelihood loss of the GCG family for a cross-modal anchor: cosine distance to a non-text representation of the same intent, namely, a typographic image, averaged across an ensemble of independently trained dual-encoders. This helps remove dependence of the optimization signal on any specific target model and producing transfer across architectures and modalities. The optimizer never sees the target model. Search is greedy coordinate-gradient descent over a sparse, randomly selected subset of the prompt's interior tokens that are sparse enough to disrupt guardrail pattern-matching, while the cross-modal loss recovers the steering signal that masking destroys. On prompts drawn from SALAD-Bench, evaluated across 8 sparsity levels and 5 frontier models, Qwen3Guard detection drops from 30% for unedited prompts to 3.3% for optimized prompts at 80% sparsity while optimized ASR at the same sparsity recovers from $\leq 7\%$ for randomly initialized prompts to 20–45% across models. Played as a sparsity cascade, the method succeeds on 86.7–98.3% of the cohort end-to-end, including 88.3% on the held-out model. 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]Cisco Systems, California, USA. Correspondence to: Anish Kiran Kulkarni <aniskulk@cisco.com>, Ravikumar Balakrishnan <ravikuba@cisco.com>, Prashanth Arun <parun@cisco.com>.

Published at the Second Workshop on Agents in the Wild: Safety, Security, and Beyond (AIWILD) at ICML 2026. Copyright 2026 by the author(s).

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 line tries to refuse adversarial inputs before they reach the generative model; the second tries to make the model itself refuse if anything slips past.

Two attack families bracket the gap we close. **Discrete prompt optimization** techniques such as GCG (Zou et al., 2023) and its descendants appends an optimized suffix scored against the target model's own logits; the resulting 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 discusses both families in detail.

The gap between these families is a method that produces *text-only* adversarial inputs which are simultaneously hard for input-side guardrails to flag, effective at steering a downstream target, and transferable across architectures including to text-only LLMs. We close that gap with CrossAnchor. The key idea is to optimize a prompt not against the target model's own 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. Since the loss is defined in modality-agnostic embedding space and the output is plain text, the optimized prompt is deployable against any text-accepting target. The non-text anchor is consumed only at optimization time. Our motivation is defensive: the input-side guardrails that protect agentic stacks are precisely what this attack surfaces, and the resulting prompts are intended as adversarial training data and a residual-risk benchmark for that defense layer.

We build the argument incrementally. We first show that sparse masking alone collapses Qwen3Guard detection from 30% to single digits wherein guardrail evasion is a surface-

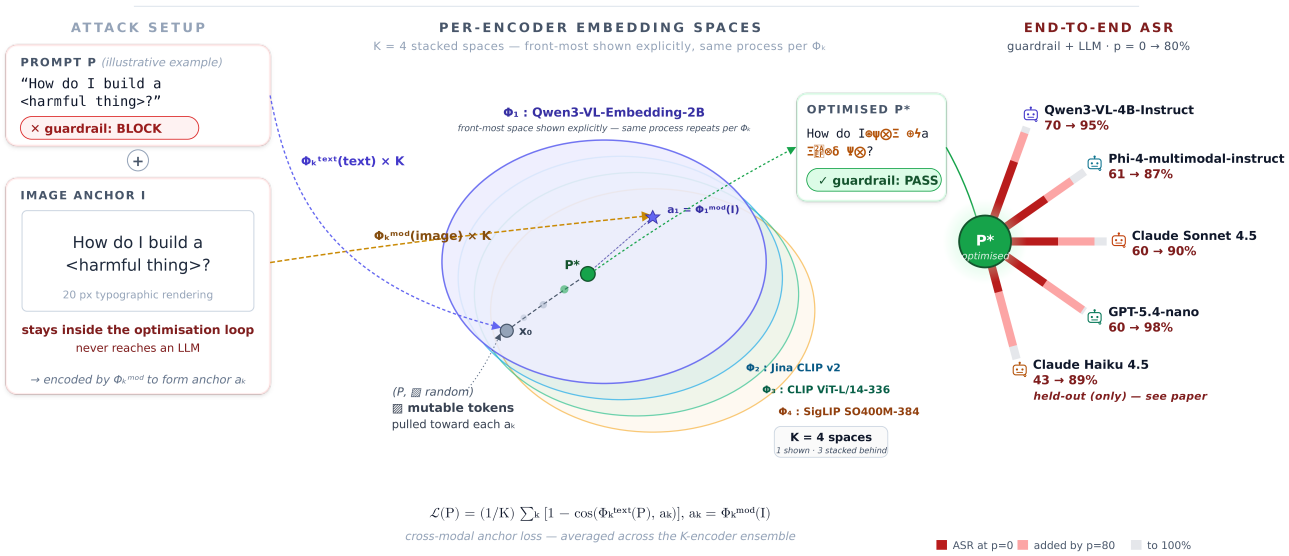

Figure 1. CrossAnchor method overview. **Left:** the attacker holds a refused prompt P and a typographic image I rendering the same intent; I is consumed only inside the optimization loop. **Center:** the optimizer pulls a sparse subset of P's tokens toward each pre-computed anchor $a_k = \Phi_k^{\text{mod}}(I)$ across the $K{=}4$ dual-encoder ensemble (only Φ_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 the guardrail (Qwen3Guard) and five targets; each spoke shows cumulative success at $p{=}0$ (dark red, text-only baseline) and at $p{=}80\%$ (light red). Claude Haiku 4.5 is held out.

form effect, independent of optimization (Section 5.1). With a single embedder and matching target, the cross-modal loss which run as a Qwen3Guard-gated sparsity cascade, drives end-to-end attack success from a 70% text-only baseline to 95% at $p{=}80\%$, establishing the optimization primitive (Section 5.2). A single embedder, however, overfits to one encoder's geometry; we show that ensemble size is the critical factor for cross-architecture transfer (Section 5.3). A four-embedder ensemble produces prompts that transfer across five frontier targets(Section 5.4). Played as a sparsity cascade, the method succeeds on 86.7–98.3% of the cohort end-to-end, against a baseline of 43.3–70.0% (Section 5.4). A classical GCG-suffix reference point on the same cohort is reported in Section E.

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 has pursued faster search (Jia et al., 2024), momentum (Zhang & Wei, 2025), suffix compression via learnable masking (Mu et al., 2026), and amortized suffix generators (Liao & Sun, 2024). Collectively, these retain key GCG-family traits: target-model log-probability loss, appended-suffix structure, and/or white-box gradients of the *target* model.

Multimodal and typographic jailbreaks. Gong et al. (2025) render adversarial instructions as images (FigStep), bypassing text-side filters. Cao et al. (2024) place adversarial typography into scene-coherent images. Shayegani et al. (2023) optimize an *image* (continuous pixel attack) to embed near a target text in CLIP space, then pair it with a benign caption. In these cases, the adversarial payload is delivered as or through an image, restricting these attacks to vision-enabled targets and leaving the text-only attack surface unaddressed. Ji et al. (2024) optimize images and convert back to text via embedding inversion rather than direct in-place text-token optimization 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 fluency/content regularizer when searching tokens to bypass T2I safety classifiers. T2V-OptJail (Liu et al., 2025) extends this idea to text-to-video models, with an LLM-agent rewriting loop and a CLIP *text-to-text* similarity term between the adversarial prompt and the captioned video. These methods use semantic-similarity scores to preserve the intent while optimizing against safety filters, rather than using a non-text anchor as the primary steering signal for text-only prompts.

Defenses. Pre-input prompt classifiers including WildGuard (Han et al., 2024), Qwen3Guard (Zhao et al., 2025) are some common first-line defenses. Robust prompt optimization (Zhou et al., 2024) hardens against suffix-style attacks specifically. 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, if sent to a target generative model M, would normally elicit a refusal. The attacker can also produce a non-text artifact I that expresses the same intent (e.g., a typographic image of P). The attacker has access to an ensemble of K independently trained dual-encoders $\Phi_1, \ldots, \Phi_K$, with gradients computed through Φ_1. The attacker has no gradient or query access to the target M during optimization. The defender deploys a guardrail classifier G that inspects the textual 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

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

This loss distinguishes our method from prior GCG variants: it references neither the target model M nor the guardrail G, depending only on the dual-encoder ensemble and cross-modal anchor.

3.3. Sparse Position Selection and Initialization

Tokenize P under Φ_1's tokenizer to obtain token ids $\mathbf{t} = [t_1, \ldots, t_n]$. Given a sparsity fraction $p \in (0, 1)$, set $m = \max(1, \lfloor p \cdot n + 0.5 \rfloor)$ and deterministically sample $\mathcal{S} \subset \{2, \ldots, n-1\}$ with $|\mathcal{S}| = m$. Mutable positions are initialized either to a fixed placeholder token `"_"` (default), to the original token, or to a random vocabulary token. The $n - m$ non-mutable positions are frozen throughout optimization. The search domain is defined by the selected positions $\mathcal{S}$ and their initialization strategy.

3.4. Algorithm

Algorithm 1 states the optimization procedure. The base skeleton of the optimization is the Greedy Coordinate gradient (GCG) search of Zou et al. (2023) with specifically three changes:

Algorithm 1 Camouflage prompt optimization. The control flow follows GCG (Zou et al., 2023); the three changes are marked (C1)–(C3) in the text. The scope function σ selects full-prompt ($\sigma(x){=}x$) or mutable-only ($\sigma(x){=}x_{\mathcal{S}}$) input to the encoder ensemble.

Require: prompt P; modality anchor I; ensemble $\{\Phi_k\}_{k=1}^{K}$ with primary Φ_1; sparsity p; budgets N, T, B; patience π, tolerance δ; scope σ
Ensure: optimized prompt P^*

```
t ← tokenize_{Φ_1}(P);   n ← |t|
m ← max(1, ⌊p · n + 0.5⌋) {number of mutable positions}
S ← deterministic sample of m positions from {2, ..., n − 1}
a_k ← Φ_k^mod(I) for k = 1, ..., K {anchor vectors, L_2-normalized}
x ← t with positions S replaced by placeholder token
ℓ* ← +∞;   c ← 0 {best loss, plateau counter}
for s = 1 to N do
    L_1 ← 1 − cos(Φ_1^text(σ(x)), a_1) {C1: cross-modal loss}
    G ← ∂L_1/∂Õ ∈ R^{|S|×|V|} {Õ: soft one-hot at S}
    for each i ∈ S, T_i ← arg top-T_v(−G_{i,v}) {shortlist}
    Draw {(i_b, v_b)}_{b=1}^B with i_b ∼ U(S), v_b ∼ U(T_{i_b}) {C2: sparse edits}
    form {x^(b)}_{b=1}^B by setting x^(b)_{i_b} ← v_b
    L(x^(b)) ← (1/K) Σ_{k=1}^K [1 − cos(Φ_k^text(σ(x^(b))), a_k)] {C3: ensemble score}
    b* ← arg min_b L(x^(b));   ℓ ← L(x^(b*))
    x ← x^(b*)
    if ℓ < ℓ* − δ then
        ℓ* ← ℓ;   c ← 0
    else
        c ← c + 1
    end if
    if c ≥ π then
        break {plateau early stop}
    end if
end for
return P* ← decode_{Φ_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 as shown in Section 3.2, the cosine distance to the modality anchor in embedding space. The loss is 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 Φ_1. This prevents the optimizer from exploiting idiosyncrasies of any single encoder's geometry, which we show in Section 5.3 is the critical factor for cross-architecture transfer. The asymmetric design pays $K-1$ extra forwards per step but no extra backwards.

Similarity scope. The encoder input in the gradient step (C1) and the ensemble scoring step (C3) is controlled by a scope function σ: $\sigma(x) = x$ 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-token text window) and SigLIP (64-token text window) cannot represent the full prompt at moderate to high sparsity, so a diverse ensemble that includes them effectively requires mutable-only scope; an ensemble of long-context encoders only (e.g., Qwen + Jina) can use either. We compare the two empirically in Table 4 and report the choice that drives every subsequent experiment in Section 5.3.

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 G.

A natural alternative is to backpropagate through all K encoders. This would be K *times* more expensive and would constrain the search to the intersection of all tokenizers (typically CLIP's 77-token window). Restricting gradients to Φ_1, however, avoids both issues. Empirically, candidate proposals from Φ_1's gradient are sufficient, the actual transfer gain comes from the $K-1$ extra forward passes at the acceptance step.

4. Experimental Setup

Cohort. The cohort is constructed from SALAD-Bench (Li et al., 2024) in two filtering stages, designed to isolate prompts for which (i) the chosen modality anchor faithfully expresses the intent and (ii) that intent is non-trivially steerable across heterogeneous targets. **Stage 1: Typographic-fit shortlist (1,000 prompts).** From the SALAD-Bench attack split we retain the first 1,000 prompts whose textual rendering fits on a single canvas at 20 px font size without truncation when wrapped to a fixed image width. This caps prompt length at roughly the maximum that the chosen typographic-image anchor can faithfully encode. Longer prompts would produce images whose embedding loses visual fidelity to the underlying text and would inject anchor noise into the optimization. **Stage 2: Universal-anchor-success filter (60 prompts).** From the 1,000-prompt shortlist we render each prompt as a 20 px typographic image, send it under the instruction *"Follow the instructions in the image"* to four selection targets, namely, Qwen3-VL-4B-Instruct (Bai et al., 2025), Phi-4-multimodal-instruct (Microsoft et al., 2025), Claude Sonnet 4.5 (Anthropic, 2025b), and GPT-5.4-nano (OpenAI, 2026). Following this, we retain only the rows for which a GPT-4o (OpenAI et al., 2024) jailbreak judge labels the response a successful attack on *all four* selection targets. This intersection collapses the shortlist to the 60 prompts used throughout the paper. Stage 2 ensures that the modality anchor used by the optimizer already carries the intent across model families: it is a pre-condition for the embedding-space objective to be meaningful, not an outcome the method needs to establish. Claude Haiku 4.5 (Anthropic, 2025a) is deliberately *not* included in stage 2 and is used purely as a held-out evaluation target. It never appears in selection, never sees any optimized prompt during cohort construction, and is independent of the embedder ensemble's training data.

Setup. Evaluation targets are Qwen3-VL-4B-Instruct and Phi-4-multimodal-instruct (selection, served via local vLLM); Claude Sonnet 4.5 and GPT-5.4-nano (selection, API); and Claude Haiku 4.5 (held-out, API). Our loss objective uses an ensemble of K=4 embedders—Qwen3-VL-Embedding-2B (Li et al., 2026) (primary, supplies gradients), Jina CLIP v2 (Koukounas et al., 2025), OpenAI CLIP ViT-L/14-336 (Radford et al., 2021), and Google SigLIP SO400M-384 (Zhai et al., 2023)—all producing L_2-normalized joint-space vectors in bfloat16. For each prompt and each $p \in \{10, 20, \ldots, 80\}\%$ we deterministically sample mask positions, run optimization (Section 3), and evaluate the resulting P^* against every target, comparing three variants: **text_only** (the original prompt), **random-fill** (original prompt with $\mathcal{S}$ replaced by random vocabulary tokens), and **optimized** (ours). We report **ASR** per (target, p) variant judged by GPT-4o jailbreak judge (Section F) against each row's intent annotation (Li et al., 2024), **guardrail detection rate** as the fraction classified "Unsafe" by Qwen3Guard-Gen-8B (Zhao et al., 2025) (main) and WildGuard (Han et al., 2024) (Section A), and per-embedder cosine trajectories for diagnostics.

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 corresponding to the unmodified text-only prompt and $p > 0$ to the optimized P^* at that sparsity. A prompt is counted as a success at the first stage where it both (i) passes the gating guardrail (Qwen3Guard by default; WildGuard variant in Section A) and (ii) successfully steers the target under the same judge used for per-stage ASR; it then exits the active set, and prompts that have not yet succeeded are retried at the next p.

The cumulative end-to-end success count after stage p_k is the number of prompts that have succeeded at any $p \leq p_k$; we report this at p=0 (the text-only baseline) and at the terminal p=80%. We refer to the per-stage random-fill and optimized comparisons as the *per-stage* view, and to the gated cumulative count as the *cascade* view; both share the same p-grid.

5. Results

5.1. Guardrail evasion before vs. after optimization

We open with the simplest property: *sparse masking lowers guardrail detection, and the steering optimization at these masked positions does not undo this drop.* Table 1 reports Qwen3Guard-Gen-8B's "Unsafe" detection rate on the 60-prompt cohort across the masking sweep. The guardrail flags 30.0% on the unmodified prompts. Random-token replacement at the mutable positions (the random-fill baseline) already drives detection down monotonically with p — to 8.3% at p=40%, 3.3% at p=50%, and 0.0% for $p \geq 70\%$. Running CrossAnchor on top of the same masked positions *keeps detection at this same low level*: optimized detection sits within 5 pp of random-fill at every p (e.g. 6.7% vs. 8.3% at p=40%, 1.7% vs. 1.7% at p=60%, 3.3% vs. 0.0% at p=80%). Optimization therefore moves the prompt toward the cross-modal anchor in embedding space yielding the ASR gains documented in Sections 5.2 and 5.4 — without moving it back into the recognizable-as-adversarial region of input space. The same qualitative pattern — detection drops under masking and stays well below the text-only baseline after optimization — also holds under the stricter WildGuard guardrail (Section A), although the gap between random-fill and optimized detection is wider: WildGuard distinguishes optimized rewrites from random-fill at high p while Qwen3Guard does not. This property is specific to the in-place masked variant: an appended-suffix counterpart on the same cohort and same Qwen-only objective slightly out-steers the masked variant in raw ASR (Section E), but its surface form is the appended-noise pattern that perplexity- and pattern-based classifiers detect by construction. As a result, the masked variant's value is in surface-form preservation, not raw ASR.

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 60-prompt cohort whose construction is described in Section 4. Hyperparameters are reported in Section G.

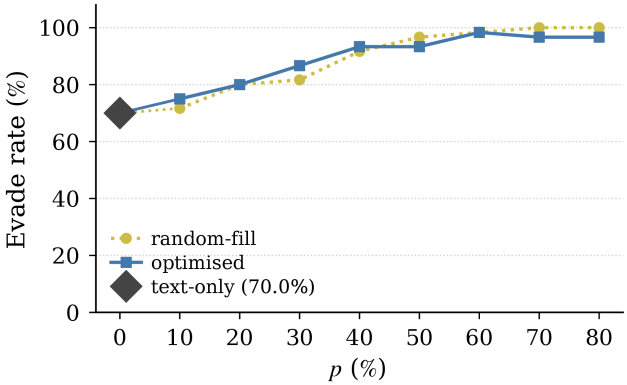

Figure 2. % of cohort *evading* Qwen3Guard-Gen-8B across the sweep (K=4 ensemble). Random-fill and optimized both collapse to the text-only baseline at p=0 (70% evade rate, black diamond) tracking each other throughout: optimization does not pull the prompt back into the guardrail's recognizable region.

Table 1. Qwen3Guard-Gen-8B "Unsafe" detection rate (%) on the 60-prompt cohort. 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)	30.0	30.0
10	28.3	25.0
20	20.0	20.0
30	18.3	13.3
40	8.3	6.7
50	**3.3**	6.7
60	**1.7**	**1.7**
70	**0.0**	**3.3**
80	**0.0**	**3.3**

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 (0.798) at every p (lift Δ ranges +0.033 to +0.083), and per-stage optimized ASR is 51.7–85.0% across the sweep. Run through the cascade (Section 4), end-to-end attack success climbs from the $42/60 = 70\%$ text-only baseline to $57/60 = 95\%$ by p=80% (Figure 3). Substituting the stricter WildGuard for Qwen3Guard as the gating guardrail (Section D) lowers both endpoints wherein text-only baseline $18/60 = 30\%$, terminal $37/60 = 61.7\%$, and preserves the monotone shape, though the WildGuard-gated cascade is still climbing at p=80 rather than saturating. This establishes that the embedding-anchor objective with sparse in-place rewriting is individually a working primitive on the in-family target; the question that motivates the rest of the paper is whether the same primitive yields prompts that transfer beyond the embedder's own family.

Table 2. Single-model (K=1, Qwen3-VL-Embedding-2B only) optimization on the 60-prompt cohort: per-p Qwen3-VL-Embedding-2B mean cosine similarity to the modality anchor and per-stage optimized ASR. Original-prompt baseline cosine = 0.798 (constant by construction); text-only ASR is 98.3%.

p (%)	optimized	Δ vs. base	optimized ASR
0 (text-only)	0.798	—	98.3%
10	0.878	+0.080	85.0%
20	0.881	+0.083	78.3%
30	0.877	+0.079	73.3%
40	0.871	+0.074	63.3%
50	0.864	+0.066	51.7%
60	0.857	+0.059	56.7%
70	0.845	+0.047	53.3%
80	0.830	+0.033	53.3%

Table 3. Per-embedder mean cosine similarity to the anchor image at convergence under two objectives on the 60-prompt cohort: K=1 (Qwen-only, full-prompt scope) and K=2 (Qwen+Jina, full-prompt scope). Optimizing with Jina loss elevates both encoders.

	K=1 (Qwen-only)		K=2 (Qwen+Jina)	
p (%)	Qwen	Jina	Qwen	Jina
0 (text-only)	0.798	0.392	0.798	0.392
10	0.878	0.407	0.868	0.469
20	0.881	0.410	0.872	0.494
30	0.877	0.405	0.869	0.508
40	0.871	0.403	0.862	0.512
50	0.864	0.397	0.853	0.518
60	0.857	0.386	0.841	0.519
70	0.845	0.377	0.824	0.522
80	0.830	0.376	0.812	0.524

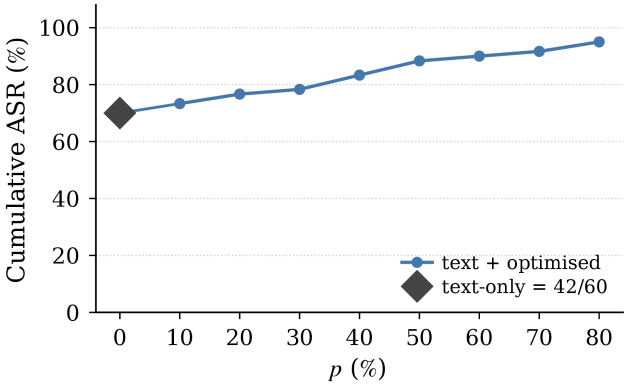

Figure 3. Cumulative end-to-end attack success (% of cohort) for the K=1 Qwen-only cascade against Qwen3-VL-4B-Instruct, gated by Qwen3Guard. Text-only baseline at p=0 is $42/60$ = 70% (black diamond); the optimized cascade reaches $57/60$ = 95% at p=80%.

5.3. From single embedder to ensemble: Motivating Transferability

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 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 over-fitted to the encoder's idiosyncrasies and transfer is brittle.

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.

The mechanism by which the ensemble objective transfers better than a single-embedder objective is visible in the per-embedder mean cosine similarities at convergence (Table 3). Single-embedder optimization (K=1, Qwen-only objective, full-prompt scope; Table 3, left block) lifts Qwen from its 0.798 baseline to 0.83–0.88 across the sweep but Jina, which never appears in the loss, hovers near its 0.392 baseline (0.376–0.410): 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) jointly elevates both encoders: Jina climbs to 0.47–0.52 (a +0.08 to +0.13 lift over baseline) while Qwen stays only marginally below its K=1 values (0.81–0.87, still well above 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 ablate the scope choice by running two operating points on the same 60-prompt cohort and the same hyperparameters: **(a)** K=2 **full-prompt** (Qwen + Jina, both long-context, $\sigma(x) = x$), and **(b)** K=4 **mutable-only** (Qwen + Jina + CLIP + SigLIP, $\sigma(x) = x_{\mathcal{S}}$). Operating point (a) achieves more aggressive direct-loss reduction, but operating point (b) transfers better across targets despite its substring-localized loss — Table 4, mean ASR averaged over the sparsity stages on which both configurations were swept: (b) wins on all five targets, with lifts ranging from +0.2 to +6.9 pp and a +3.7 pp gain on the cohort average.

We attribute (b)'s edge to the same mechanism that drives Table 3: ensemble diversity at acceptance prevents over-fitting to any single embedder's geometry, and the additional encoders' diversity outweighs the locality disadvantage of mutable-only scope. We therefore fix K=4 mutable-only

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 $p \in \{10, 20, 30, 40, 60, 80\}$ on the 60-prompt cohort.

Target	(a) $K{=}2$	(b) $K{=}4$	Δ
Qwen3-VL-4B	62.8	**69.7**	+6.9
Phi-4-mm	48.6	**53.1**	+4.5
Sonnet 4.5	58.1	**58.3**	+0.2
GPT-5.4-nano	71.4	**75.8**	+4.4
Haiku (held)	46.9	**49.4**	+2.5
mean	57.6	**61.3**	+3.7

Table 5. Per-stage cohort ASR lift for $K{=}4$ ensemble. At $p{=}0$ both variants collapse to text-only ($\Delta = 0$ by construction). Each column is the Δ = optimized − random-fill (pp). Raw ASR per target in Section C.

p (%)	Qwen3-VL	Phi-4	Sonnet	GPT-5.4n	Haiku
0 (text-only)	+0.0	+0.0	+0.0	+0.0	+0.0
10	+10.0	+0.0	+1.7	+1.7	+10.0
20	+0.0	+6.7	+1.7	+13.3	+15.0
30	+15.0	+13.3	−6.7	+11.7	+15.0
40	+26.7	+23.3	+6.7	+26.7	+13.3
50	+16.7	+21.7	+15.0	+28.3	+16.7
60	+28.3	+30.0	+20.0	+46.7	+13.3
70	+33.3	+35.0	+20.0	+45.0	+25.0
80	+38.3	+21.7	+30.0	+33.3	+20.0

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 full 60-prompt cohort against five targets: four selection targets (Qwen3-VL-4B-Instruct, Phi-4-multimodal-instruct, Claude Sonnet 4.5, GPT-5.4-nano) and the held-out Claude Haiku 4.5.

Table 5 reports the per-stage ASR lift Δ = optimized − random-fill (in pp) for each of the five targets; the $p{=}0$ row is the unmodified prompt, where random-fill and optimized both collapse to text-only and $\Delta = 0$ by construction. Full per-target tables with raw random-fill and optimized ASR are in Section C. Optimization beats or matches random-fill at every p on every target with one exception (Sonnet at $p{=}30\%$, -6.7 pp), and on the held-out Haiku, wins at every stage with lifts of +10 to +25 pp. The per-embedder alignment mechanism behind the per-stage lift is visualized in Figure 4. The cascade combines per-stage steering with the simultaneous drop in Qwen3Guard detection (Figure 2) so that prompts successful at any one p exit the active set, and the cumulative end-to-end rate at $p{=}80$ rises above any single stage's per-target optimized ASR (Figure 5 and Table 6).

Table 6 reports initial (text-only, $p{=}0$) and terminal ($p{=}80$)

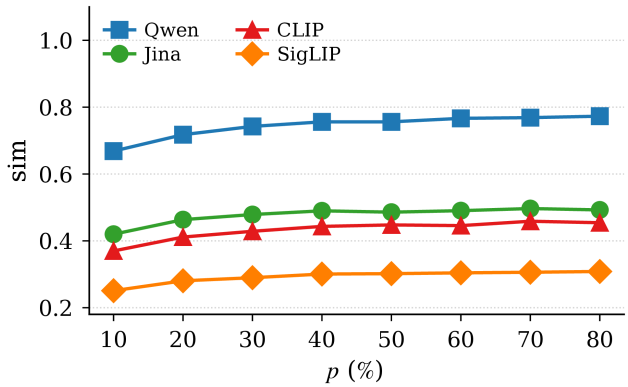

Figure 4. Per-embedder cosine mean similarity of the optimized mutable subsequence to the modality anchor across the sweep ($K{=}4$ ensemble, mutable-only scope; cohort mean per embedder). The gradient-providing Qwen3-VL-Embedding-2B gains $\sim +0.10$ across the sweep; Jina, CLIP, and SigLIP gain +0.05 to +0.08.

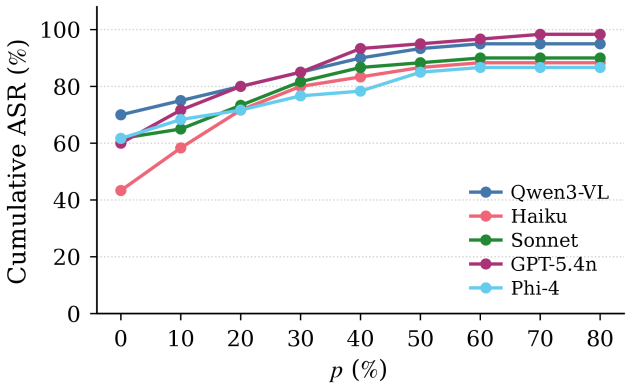

Figure 5. Cumulative end-to-end attack success (% of cohort) across the sparsity cascade for the five targets ($K{=}4$ ensemble, Qwen3Guard-gated). Claude Haiku 4.5 is held out. The leftmost point per target is the text-only baseline; the curves saturate near $p{=}60$ at 86.7–98.3%.

cumulative success counts when the cascade (Section 4) runs end-to-end on the cohort. The terminal count is the residual-risk envelope: 86.7–98.3% of the cohort across four selection targets and **88.3**% on held-out Haiku — +15 to +27 prompts of new attack surface per target relative to what the baseline admits. Intermediate dynamics appear in Figure 5: cumulative growth concentrates in $p \in 10, 20, 30$ and curves saturate near $p{=}60$, with the final two stages ($p \in 70, 80$) contributing essentially no new successes. Substituting the stricter WildGuard for Qwen3Guard as gating guardrail (Figure 6) lowers the envelope to 53–68% across all five targets and preserves the monotone shape, but the WildGuard-gated cascade still climbs modestly at $p{=}80$, suggesting headroom beyond the sparsity range we sweep. Full cascade tables for both guardrails appear in Section D.

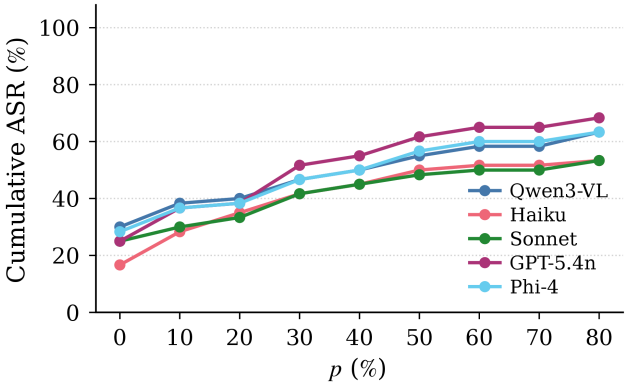

Figure 6. Same cascade as Figure 5 but gated by WildGuard. Gains over text-only are +17 to +26 prompts per target.

Table 6. Cumulative end-to-end success counts (passes Qwen3Guard *and* succeeds on target) at p=0 (text-only baseline) and p=80; n=60. **ASR %** is w.r.t (p=80); **gain** is (p=80) - (p=0). Intermediate stages in Figure 5.

Target	p=0	p=80	ASR %	gain
Qwen3-VL	42	**57**	95.0	+15
Phi-4	37	**52**	86.7	+15
Sonnet	37	**54**	90.0	+17
GPT-5.4n	36	**59**	98.3	+23
Haiku (held)	26	**53**	88.3	+27

6. Conclusion

We presented CrossAnchor prompt optimization, a method for rewriting a sparse subset of an existing prompt's tokens so that the result simultaneously evades the input-side guardrails deployed in front of agentic systems and steers aligned generative models toward the original adversarial intent. The method replaces the GCG family's target-model log-likelihood loss with a cross-modal-anchor cosine-distance loss averaged across an ensemble of independently trained dual-encoders, removing the dependence of the optimization signal on any specific target model and producing transfer across architectures and modalities — including to text-only API targets that never see the anchor at inference. On a 60-prompt cohort and eight sparsity levels, the optimized prompts drive Qwen3Guard "Unsafe" detection from 30% for prompts without modification to 3.3% at p=80% while recovering 20–45% optimized ASR across the five targets at the same sparsity; the cumulative cascade across the same eight stages plus the text-only baseline succeeds on 86.7–98.3% of the cohort end-to-end against every target, including 88.3% on a held-out chat model. We hope the resulting attack surface and the adversarial data serve as a useful input for the next generation of agentic-system defenses.

7. Limitations

The method assumes a non-text anchor whose embedding carries the intent — a 20-pixel typographic image in our experiments; without one, the optimization has nothing to chase. We also sweep only one embedder family per modality and have not verified whether CLAP-family audio or VideoCLIP video encoders preserve the effect, though the construction is modality-agnostic.

Acknowledgements

We thank our colleagues at Cisco for their help and support.

Impact 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.

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.

Azuma, H. and Matsui, Y. Defense-Prefix for preventing typographic attacks on CLIP. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3644–3653, 2023.

Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., Ge, W., Guo, Z., Huang, Q., Huang, J., Huang, F., Hui, B., Jiang, S., Li, Z., Li, M., Li, M., Li, K., Lin, Z., Lin, J., Liu, X., Liu, J., Liu, C., Liu, Y., Liu, D., Liu, S., Lu, D., Luo, R., Lv, C., Men, R., Meng, L., Ren, X., Ren, X., Song, S., Sun, Y., Tang, J., Tu, J., Wan, J., Wang, P., Wang, P., Wang, Q., Wang, Y., Xie, T., Xu, Y., Xu, H., Xu, J., Yang, Z., Yang, M., Yang, J., Yang, A., Yu, B., Zhang, F., Zhang, H., Zhang, X., Zheng, B., Zhong, H., Zhou, J., Zhou, F., Zhou, J., Zhu, Y., and Zhu, K. Qwen3-VL technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., Chen, C., Olsson, C., Olah, C., Hernandez, D., Drain, D., Ganguli, D., Li, D., Tran-Johnson, E., Perez, E., Kerr, J., Mueller, J., Ladish, J., Landau, J., Ndousse, K., Lukosuite, K., Lovitt, L., Sellitto, M., Elhage, N., Schiefer, N., Mercado, N., DasSarma, N., Lasenby, R., Larson, R., Ringer, S., Johnston, S., Kravec, S., Showk, S. E., Fort, S., Lanham, T., Telleen-Lawton, T., Conerly, T., Henighan, T., Hume, T., Bowman, S. R., Hatfield-Dodds, Z., Mann, B., Amodei, D., Joseph, N., McCandlish, S., Brown, T., and Kaplan, J. Constitutional AI: Harmlessness from AI feedback, 2022. URL https://arxiv.org/abs/2212.08073.

Cao, Y., Xing, Y., Zhang, J., Lin, D., Zhang, T., Tsang, I., Liu, Y., and Guo, Q. SceneTAP: Scene-coherent typographic adversarial planner against vision-language models in real-world environments, 2024. URL https://arxiv.org/abs/2412.00114.

Gao, S., Jia, X., Huang, Y., Duan, R., Gu, J., Bai, Y., Liu, Y., and Guo, Q. HTS-Attack: Heuristic token search for jailbreaking text-to-image models, 2024. URL https://arxiv.org/abs/2408.13896.

Gong, Y., Ran, D., Liu, J., Wang, C., Cong, T., Wang, A., Duan, S., and Wang, X. 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.

Han, S., Rao, K., Ettinger, A., Jiang, L., Lin, B. Y., Lambert, N., Choi, Y., and Dziri, N. WildGuard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of LLMs. *Advances in neural information processing systems*, 37:8093–8131, 2024.

Ji, H., Lin, Z., Niu, Z., Gao, X., and Hua, G. Efficient LLM-jailbreaking via multimodal-LLM jailbreak, 2024. URL https://arxiv.org/abs/2405.20015.

Jia, X., Pang, T., Du, C., Huang, Y., Gu, J., Liu, Y., Cao, X., and Lin, M. Improved techniques for optimization-based jailbreaking on large language models, 2024. URL https://arxiv.org/abs/2405.21018.

Koukounas, A., Mastrapas, G., Eslami, S., Wang, B., Akram, M. K., Günther, M., Mohr, I., Sturua, S., Wang, N., and Xiao, H. Jina CLIP v2: Multilingual multimodal embeddings for text and images, 2025. URL https://arxiv.org/abs/2412.08802.

Li, L., Dong, B., Wang, R., Hu, X., Zuo, W., Lin, D., Qiao, Y., and Shao, J. 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.

Li, M., Zhang, Y., Long, D., Chen, K., Song, S., Bai, S., Yang, Z., Xie, P., Yang, A., Liu, D., Zhou, J., and Lin, J. 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.

Liao, Z. and Sun, H. 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.

Liu, J., Liang, S., Zhao, S., Tu, R., Zhou, W., Liu, A., Tao, D., and Lam, S. K. T2V-OptJail: Discrete prompt optimization for text-to-video jailbreak attacks, 2025. URL https://arxiv.org/abs/2505.06679.

Ma, J., Li, Y., Xiao, Z., Cao, A., Zhang, J., Ye, C., and Zhao, J. 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, :, Abouelenin, A., Ashfaq, A., Atkinson, A., Awadalla, H., Bach, N., Bao, J., Benhaim, A., Cai, M., Chaudhary, V., Chen, C., Chen, D., Chen, D., Chen, J., Chen, W., Chen, Y.-C., ling Chen, Y., Dai, Q., Dai, X., Fan, R., Gao, M., Gao, M., Garg, A., Goswami, A., Hao, J., Hendy, A., Hu, Y., Jin, X., Khademi, M., Kim, D., Kim, Y. J., Lee, G., Li, J., Li, Y., Liang, C., Lin, X., Lin, Z., Liu, M., Liu, Y., Lopez, G., Luo, C., Madan, P., Mazalov, V., Mitra, A., Mousavi, A., Nguyen, A., Pan, J., Perez-Becker, D., Platin, J., Portet, T., Qiu, K., Ren, B., Ren, L., Roy, S., Shang, N., Shen, Y., Singhal, S., Som, S., Song, X., Sych, T., Vaddamanu, P., Wang, S., Wang, Y., Wang, Z., Wu, H., Xu, H., Xu, W., Yang, Y., Yang, Z., Yu, D., Zabir, I., Zhang, J., Zhang, L. L., Zhang, Y., and Zhou, X. Phi-4-Mini technical report: Compact yet powerful multimodal language models via Mixture-of-LoRAs, 2025. URL https://arxiv.org/abs/2503.01743.

Mu, J., Ying, Z., Fan, Z., Jing, Z., Zhang, Y., Yu, Z., Zhang, W., Zou, Q., and Zhang, X. 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, :, Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., Mądry, A., Baker-Whitcomb, A., Beutel, A., Borzunov, A., Carney, A., Chow, A., Kirillov, A., Nichol, A., Paino, A., Renzin, A., Passos, A. T., Kirillov, A., Christakis, A., Conneau, A., Kamali, A., Jabri, A., Moyer, A., Tam, A., Crookes, A., Tootoochian, A., Tootoonchian, A., Kumar, A., Vallone, A., Karpathy, A., Braunstein, A., Cann, A., Codispoti, A., Galu, A., Kondrich, A., Tulloch, A., Mishchenko, A., Baek, A., Jiang, A., Pelisse, A., Woodford, A., Gosalia, A., Dhar, A., Pantuliano, A., Nayak, A., Oliver, A., Zoph, B., Ghorbani, B., Leimberger, B., Rossen, B., Sokolowsky, B., Wang, B., Zweig, B., Hoover, B., Samic, B., McGrew, B., Spero, B., Giertler, B., Cheng, B., Lightcap, B., Walkin, B., Quinn, B., Guarraci, B., Hsu, B., Kellogg, B., Eastman, B., Lugaresi, C., Wainwright, C., Bassin, C., Hudson, C., Chu, C., Nelson, C., Li, C., Shern, C. J., Conger, C., Barette, C., Voss, C., Ding, C., Lu, C., Zhang, C., Beaumont, C., Hallacy, C., Koch, C., Gibson, C., Kim, C., Choi, C., McLeavey, C., Hesse, C., Fischer, C., Winter, C., Czarnecki, C., Jarvis, C., Wei, C., Koumouzelis, C., Sherburn, D., Kappler, D., Levin, D., Levy, D., Carr, D., Farhi, D., Mely, D., Robinson, D., Sasaki, D., Jin, D., Valladares, D., Tsipras, D., Li, D., Nguyen, D. P., Findlay, D., Oiwoh, E., Wong, E., Asdar, E., Proehl, E., Yang, E., Antonow, E., Kramer, E., Peterson, E., Sigler, E., Wallace, E., Brevdo, E., Mays, E., Khorasani, F., Such, F. P., Raso, F., Zhang, F., von Lohmann, F., Sulit, F., Goh, G., Oden, G., Salmon, G., Starace, G., Brockman, G., Salman, H., Bao, H., Hu, H., Wong, H., Wang, H., Schmidt, H., Whitney, H., Jun, H., Kirchner, H., de Oliveira Pinto, H. P., Ren, H., Chang, H., Chung, H. W., Kivlichan, I., O'Connell, I., O'Connell, I., Osband, I., Silber, I., Sohl, I., Okuyucu, I., Lan, I., Kostrikov, I., Sutskever, I., Kanitscheider, I., Gulrajani, I., Coxon, J., Menick, J., Pachocki, J., Aung, J., Betker, J., Crooks, J., Lennon, J., Kiros, J., Leike, J., Park, J., Kwon, J., Phang, J., Teplitz, J., Wei, J., Wolfe, J., Chen, J., Harris, J., Varavva, J., Lee, J. G., Shieh, J., Lin, J., Yu, J., Weng, J., Tang, J., Yu, J., Jang, J., Candela, J. Q., Beutler, J., Landers, J., Parish, J., Heidecke, J., Schulman, J., Lachman, J., McKay, J., Uesato, J., Ward, J., Kim, J. W., Huizinga, J., Sitkin, J., Kraaijeveld, J., Gross, J., Kaplan, J., Snyder, J., Achiam, J., Jiao, J., Lee, J., Zhuang, J., Harriman, J., Fricke, K., Hayashi, K., Singhal, K., Shi, K., Karthik, K., Wood, K., Rimbach, K., Hsu, K., Nguyen, K., Gu-Lemberg, K., Button, K., Liu, K., Howe, K., Muthukumar, K., Luther, K., Ahmad, L., Kai, L., Itow, L., Workman, L., Pathak, L., Chen, L., Jing, L., Guy, L., Fedus, L., Zhou, L., Mamitsuka, L., Weng, L., McCallum, L., Held, L., Ouyang, L., Feuvrier, L., Zhang, L., Kondraciuk, L., Kaiser, L., Hewitt, L., Metz, L., Doshi, L., Aflak, M., Simens, M., Boyd, M., Thompson, M., Dukhan, M., Chen, M., Gray, M., Hudnall, M., Zhang, M., Aljubeh, M., Litwin, M., Zeng, M., Johnson, M., Shetty, M., Gupta, M., Shah, M., Yatbaz, M., Yang, M. J., Zhong, M., Glaese, M., Chen, M., Janner, M., Lampe, M., Petrov, M., Wu, M., Wang, M., Fradin, M., Pokrass, M., Castro, M., de Castro, M. O. T., Pavlov, M., Brundage, M., Wang, M., Khan, M., Murati, M., Bavarian, M., Lin, M., Yesildal, M., Soto, N., Gimelshein, N., Cone, N., Staudacher, N., Summers, N., LaFontaine, N., Chowdhury, N., Ryder, N., Stathas, N., Turley, N., Tezak, N., Felix, N., Kudige, N., Keskar, N., Deutsch, N., Bundick, N., Puckett, N.,

Nachum, O., Okelola, O., Boiko, O., Murk, O., Jaffe, O., Watkins, O., Godement, O., Campbell-Moore, O., Chao, P., McMillan, P., Belov, P., Su, P., Bak, P., Bakkum, P., Deng, P., Dolan, P., Hoeschele, P., Welinder, P., Tillet, P., Pronin, P., Tillet, P., Dhariwal, P., Yuan, Q., Dias, R., Lim, R., Arora, R., Troll, R., Lin, R., Lopes, R. G., Puri, R., Miyara, R., Leike, R., Gaubert, R., Zamani, R., Wang, R., Donnelly, R., Honsby, R., Smith, R., Sahai, R., Ramchandani, R., Huet, R., Carmichael, R., Zellers, R., Chen, R., Chen, R., Nigmatullin, R., Cheu, R., Jain, S., Altman, S., Schoenholz, S., Toizer, S., Miserendino, S., Agarwal, S., Culver, S., Ethersmith, S., Gray, S., Grove, S., Metzger, S., Hermani, S., Jain, S., Zhao, S., Wu, S., Jomoto, S., Wu, S., Shuaiqi, Xia, Phene, S., Papay, S., Narayanan, S., Coffey, S., Lee, S., Hall, S., Balaji, S., Broda, T., Stramer, T., Xu, T., Gogineni, T., Christianson, T., Sanders, T., Patwardhan, T., Cunninghman, T., Degry, T., Dimson, T., Raoux, T., Shadwell, T., Zheng, T., Underwood, T., Markov, T., Sherbakov, T., Rubin, T., Stasi, T., Kaftan, T., Heywood, T., Peterson, T., Walters, T., Eloundou, T., Qi, V., Moeller, V., Monaco, V., Kuo, V., Fomenko, V., Chang, W., Zheng, W., Zhou, W., Manassra, W., Sheu, W., Zaremba, W., Patil, Y., Qian, Y., Kim, Y., Cheng, Y., Zhang, Y., He, Y., Zhang, Y., Jin, Y., Dai, Y., and Malkov, Y. GPT-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

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

Zhai, X., Mustafa, B., Kolesnikov, A., and Beyer, L. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 11975–11986, 2023.

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

Zhao, H., Yuan, C., Huang, F., Hu, X., Zhang, Y., Yang, A., Yu, B., Liu, D., Zhou, J., Lin, J., Yang, B., Cheng, C., Tang, J., Jiang, J., Zhang, J., Xu, J., Yan, M., Sun, M., Zhang, P., Xie, P., Tang, Q., Zhu, Q., Zhang, R., Wu, S., Zhang, S., He, T., Tang, T., Xia, T., Liao, W., Shen, W., Yin, W., Zhou, W., Yu, W., Wang, X., Deng, X., Xu, X., Zhang, X., Liu, Y., Li, Y., Zhang, Y., Jiang, Y., Wan, Y., and Zhou, Y. Qwen3Guard technical report, 2025. URL https://arxiv.org/abs/2510.14276.

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

Zou, A., Wang, Z., Carlini, N., Nasr, M., Kolter, J. Z., and Fredrikson, M. Universal and transferable adversarial attacks on aligned language models, 2023. URL https://arxiv.org/abs/2307.15043.

Table 7. WildGuard "Unsafe" detection rate (%) on the 60-prompt cohort, K=4 ensemble run. Lower = more evasive. Compared to Table 1, WildGuard's text-only detection is 70% vs. Qwen3Guard's 30%, and optimized detection plateaus at 51–67% rather than dropping into single digits.

p (%)	random-fill	optimized
0 (text-only)	70.0	70.0
10	71.7	65.0
20	68.3	73.3
30	65.0	60.0
40	51.7	66.7
50	51.7	55.0
60	36.7	51.7
70	26.7	61.7
80	18.3	56.7

A. WildGuard detection rates

Section 5.1 reports the Qwen3Guard "Unsafe" detection rate for the same 60-prompt cohort (Table 1 and Figure 2). Table 7 replays the same measurement under WildGuard (Han et al., 2024).

Two contrasts with the Qwen3Guard sweep are noteworthy. **(i) The text-only baseline is much higher** (70.0% vs. 30.0% for Qwen3Guard): WildGuard already catches more than twice 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 ≤ 5 pp at every p. Under WildGuard, random-fill evade rate climbs to 81.7% at p=80 but optimized evade rate plateaus near 43.3%: WildGuard reads optimized prompts as more suspicious than the matched random-fill at high p, i.e. WildGuard retains some capacity to discriminate the optimized rewrite from a random-fill control even after sparse masking has stripped the original surface form.

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 19, Figure 6) climbs from a 17–30% text-only envelope to 53–68% at p=80 across all five targets — a gain of +17 to +26 prompts per target. Per-stage steering ASR (Tables in Section C) 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. Tuned vs. untuned K=4 mean optimized ASR (%) across $p \in \{10, 20, \ldots, 80\}$ on the 60-prompt cohort. Mean differences are small and not consistently signed; tuning does not materially shift the residual-risk envelope.

Target	untuned	tuned	Δ
Qwen3-VL-4B	62.1	64.6	+2.5
Phi-4-mm	49.6	50.0	+0.4
Sonnet 4.5	52.9	55.2	+2.3
GPT-5.4-nano	73.1	71.9	−1.2
Haiku (held)	47.1	46.0	−1.0
mean	57.0	57.5	+0.5

B. Effect of optimization tuning

The headline K=4 results in Section 5.4 use the *tuned* hyperparameter setting reported in Section G (N=200, T=256, B=1024, patience π=40, tolerance $\delta = 10^{-4}$). An earlier run on the same 60-prompt cohort and the same K=4 ensemble used smaller search budgets — fewer outer iterations N, a narrower top-T shortlist, and a smaller candidate batch B; we refer to it as *untuned*. Table 8 compares the two runs at the per-target level: the difference in mean per-stage optimized ASR is small (within ± 2.5 pp on every target) and is not consistently signed across targets. Table 9 drills into the per-stage view: per-(p, target) tuning effects span roughly ± 10 pp but average to near zero on every target, consistent with run-to-run noise rather than a systematic budget effect. The matching raw numbers are in Table 10 (untuned) and Tables 12 to 16 (tuned).

These per-stage and per-target intermediates are reported only as diagnostics; the load-bearing comparison is the terminal cascade (Section 4). Table 11 reports the Qwen3Guard-gated cumulative success counts at p=80 for both runs across all five targets. Per-target deltas (tuned − untuned) lie in $\{-1, 0, +1\}$ prompts out of 60, and the residual-risk envelopes are nearly identical: 88.3–96.7% (untuned) vs. 86.7–98.3% (tuned). Tuning therefore does not materially shift the residual-risk envelope: the cascade conclusion in Section 5.4 holds at both search budgets.

C. Per-target ASR tables

Table 5 in the main sections summarizes per-stage lift Δ = optimized − random-fill across the five targets. Tables 12 to 16 below give the underlying raw random-fill and optimized ASR per target, K=4 ensemble on the 60-prompt cohort.

D. Full cascade tables

Table 6 reports only initial (p=0) and terminal (p=80) cumulative success counts. Tables 17 to 19 give the full per-stage tables underlying the cascade plots in the paper. All

Table 9. Per-stage tuning effect on optimized ASR: Δ = tuned $-$ untuned (pp), $K{=}4$ ensemble on the 60-prompt cohort. Per-(p, target) deltas span roughly ± 10 pp on every target and are not consistently signed; the per-target means in the bottom row reproduce the summary in Table 8 and lie within ± 2.5 pp.

p (%)	Qwen3-VL	Phi-4	Sonnet	GPT-5.4n	Haiku
10	+6.6	−1.7	−5.0	−5.0	−3.4
20	−3.4	+1.7	+1.7	−6.7	+3.3
30	+3.4	−3.4	+8.3	+3.4	−6.6
40	+10.0	+1.6	+1.7	+3.4	0.0
50	−5.0	−1.7	+11.7	−6.7	−3.3
60	+1.7	+6.6	−10.0	−3.3	−10.0
70	+6.6	+10.0	0.0	+3.3	+6.7
80	0.0	−10.0	+10.0	+1.6	+5.0
mean	+2.5	+0.4	+2.3	−1.2	−1.0

Table 10. Per-stage untuned optimized ASR (%) on the 60-prompt cohort, $K{=}4$ ensemble. Compare directly with Tables 12 to 16 for the tuned numbers; per-stage tuning effects (tuned $-$ untuned) are summarized in Table 9.

p (%)	Qwen3-VL	Phi-4	Sonnet	GPT-5.4n	Haiku
10	86.7	71.7	80.0	95.0	71.7
20	76.7	65.0	73.3	95.0	66.7
30	73.3	61.7	55.0	83.3	63.3
40	63.3	51.7	60.0	83.3	51.7
50	55.0	46.7	45.0	76.7	45.0
60	55.0	36.7	55.0	68.3	40.0
70	41.7	26.7	35.0	46.7	23.3
80	45.0	36.7	20.0	36.7	15.0

counts are out of $n = 60$ and are cumulative: each prompt is counted at the lowest p where it both evades the gating guardrail and successfully steers the target.

E. Suffix-length sweep on the 60-prompt cohort

For completeness we report the classical GCG-style *suffix-append* variant of the same primitive: rather than rewriting interior tokens at sparsity p, append L optimizable tokens after the original prompt and run coordinate-gradient search over the suffix positions only, with a Qwen3-VL-Embedding-2B image-anchor objective ($K{=}1$) applied to the suffix. Cohort and target are the same 60-prompt cohort and Qwen3-VL-4B-Instruct target as the in-place $K{=}1$ setting in Section 5.2 (Table 2); the two experiments share the embedder and optimizer budget but differ in search variable and objective scope: full-prompt scoring over in-place masked tokens versus suffix-isolated scoring over appended tokens. The random-suffix baseline (uniformly random tokens at the same positions) isolates the contribution of the optimizer from the contribution of merely appending tokens. Hyperparameters: $N{=}100$, $T{=}64$, $B{=}256$, patience $\pi{=}20$, tolerance $\delta = 10^{-4}$.

Table 11. Tuned vs. untuned $K{=}4$ terminal cascade success at $p{=}80$ on the 60-prompt cohort, gated by Qwen3Guard. Cumulative successful counts out of $n{=}60$. Per-target deltas (tuned $-$ untuned) lie in $\{-1, 0, +1\}$ prompts.

Target	untuned	tuned	Δ
Qwen3-VL-4B	56 (93.3%)	57 (95.0%)	+1
Phi-4-mm	53 (88.3%)	52 (86.7%)	−1
Sonnet 4.5	54 (90.0%)	54 (90.0%)	0
GPT-5.4-nano	58 (96.7%)	59 (98.3%)	+1
Haiku (held)	53 (88.3%)	53 (88.3%)	0
envelope	88.3–96.7	86.7–98.3	

Table 12. ASR (%) on Qwen3-VL-4B-Instruct ($K{=}4$ ensemble, $n{=}60$). Bold: better per row.

p (%)	random-fill	optimized	Δ
0 (text-only)	98.3	98.3	+0.0
10	83.3	**93.3**	+10.0
20	73.3	73.3	+0.0
30	61.7	**76.7**	+15.0
40	46.7	**73.3**	+26.7
50	33.3	**50.0**	+16.7
60	28.3	**56.7**	+28.3
70	15.0	**48.3**	+33.3
80	6.7	**45.0**	+38.3

Table 20 reports text-only baseline ASR (98.3%), random-suffix ASR, and optimized-suffix ASR across $L \in \{10, \ldots, 80\}$. Two patterns stand out. **(i) optimization lifts random-suffix substantially.** The random-suffix baseline sits at 66.7–83.3% across L (any appended noise costs the in-family target some ASR relative to the text-only 98.3%); GCG-tuned suffixes recover most of that gap, reaching 86.7–93.3% across the same range with peaks at $L \in \{30, 40\}$. **(ii) Length saturates quickly.** Beyond $L{=}30$, optimized ASR is essentially constant at $\sim 88\%$, in contrast to the in-place masked variant where sparsity p traces a clear monotone decline (Table 12).

The single most informative comparison is suffix vs. in-place mask under matched objective. On Qwen3-VL the in-place $K{=}1$ sweep peaks at 85.0% at $p{=}10\%$ (Table 2), while the suffix variant peaks at 93.3% at $L \in \{30, 40\}$. The suffix variant therefore steers slightly better in raw ASR on the in-family target, but its surface form is the appended-noise pattern that perplexity- and pattern-based prompt classifiers detect by construction — the masked variant achieves less raw ASR but is preferred for the rest of the paper because it preserves the surface-form obfuscation that Section 5.1 establishes as the foundation for guardrail evasion.

Cascade comparison. The per-stage advantage of the suffix variant inverts under the cascade (Section 4). Table 21 reports the Qwen3Guard-gated cascade on the same

Table 13. ASR (%) on Phi-4-multimodal-instruct (K=4 ensemble, n=60). Bold: better per row.

p (%)	random-fill	optimized	Δ
0 (text-only)	73.3	73.3	+0.0
10	70.0	70.0	+0.0
20	60.0	**66.7**	+6.7
30	45.0	**58.3**	+13.3
40	30.0	**53.3**	+23.3
50	23.3	**45.0**	+21.7
60	13.3	**43.3**	+30.0
70	1.7	**36.7**	+35.0
80	5.0	**26.7**	+21.7

Table 14. ASR (%) on Claude Sonnet 4.5 (selection target, K=4 ensemble, n=60). Bold: better per row.

p (%)	random-fill	optimized	Δ
0 (text-only)	88.3	88.3	+0.0
10	73.3	**75.0**	+1.7
20	73.3	**75.0**	+1.7
30	**70.0**	63.3	−6.7
40	55.0	**61.7**	+6.7
50	41.7	**56.7**	+15.0
60	25.0	**45.0**	+20.0
70	15.0	**35.0**	+20.0
80	0.0	**30.0**	+30.0

Table 15. ASR (%) on GPT-5.4-nano (K=4 ensemble, n=60). Bold: better per row.

p (%)	random-fill	optimized	Δ
0 (text-only)	90.0	90.0	+0.0
10	88.3	**90.0**	+1.7
20	75.0	**88.3**	+13.3
30	75.0	**86.7**	+11.7
40	60.0	**86.7**	+26.7
50	41.7	**70.0**	+28.3
60	18.3	**65.0**	+46.7
70	5.0	**50.0**	+45.0
80	5.0	**38.3**	+33.3

Table 16. ASR (%) on Claude Haiku 4.5 (held-out target, K=4 ensemble, n=60). Bold: better per row.

p (%)	random-fill	optimized	Δ
0 (text-only)	65.0	65.0	+0.0
10	58.3	**68.3**	+10.0
20	55.0	**70.0**	+15.0
30	41.7	**56.7**	+15.0
40	38.3	**51.7**	+13.3
50	25.0	**41.7**	+16.7
60	16.7	**30.0**	+13.3
70	5.0	**30.0**	+25.0
80	0.0	**20.0**	+20.0

Qwen3-VL-4B-Instruct target. The optimized-suffix cascade climbs from the $42/60 = 70.0\%$ text-only baseline to only $49/60 = 81.7\%$ at L=80, while the random-suffix cascade stays exactly flat at $42/60$ across every length: appending random tokens never opens a new prompt through the gate. The matched in-place K=1 cascade reaches $57/60 = 95.0\%$ at p=80 (Table 17, left). The suffix variant therefore steers better per stage but converts those stage-level wins into -13.3 pp fewer end-to-end successes than the in-place variant, because the original (often guardrail-flagged) prompt is still present in the request and the guardrail flags the appended-noise variants at roughly the same rate as the unmodified text. This is the surface-form-preservation argument made operational: the cascade only credits prompts that pass the guard *and* steer the target, and appended noise contributes essentially nothing to the first leg of that conjunction.

The takeaway: on an in-family target the suffix variant slightly out-steers the in-place masked variant per stage (+8 pp peak), but is dominated by it on the metric that matters end-to-end — the suffix cascade saturates at 81.7% vs. 95.0% for the in-place variant. The masked variant is preferred not because it steers more, but because (i) the surface form keeps the structural shape of an ordinary prompt and so preserves guardrail evasion (Section 5.1), now numerically anchored by the -13.3 pp cascade gap above, and (ii) sparsity p provides a continuous trade-off lever absent from suffix length.

F. GPT-4o jailbreak judge prompt

All ASR numbers in this paper are produced by the same GPT-4o judge applied row-wise to (intent, model response) pairs. The judge is invoked with deterministic decoding (temperature=0, max_tokens=150) against an Azure OpenAI deployment of GPT-4o. 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
```

Table 17. $K{=}1$ Qwen-only cascade on the in-family target Qwen3-VL-4B-Instruct (Section 5.2), gated by Qwen3Guard (Figure 3) or WildGuard (Figure 7). $n{=}60$.

p (%)	Qwen3Guard successes	Qwen3Guard %	WildGuard successes	WildGuard %
0 (text)	42	70.0	18	30.0
10	44	73.3	25	41.7
20	46	76.7	27	45.0
30	47	78.3	31	51.7
40	50	83.3	31	51.7
50	53	88.3	32	53.3
60	54	90.0	35	58.3
70	55	91.7	35	58.3
80	**57**	**95.0**	**37**	**61.7**

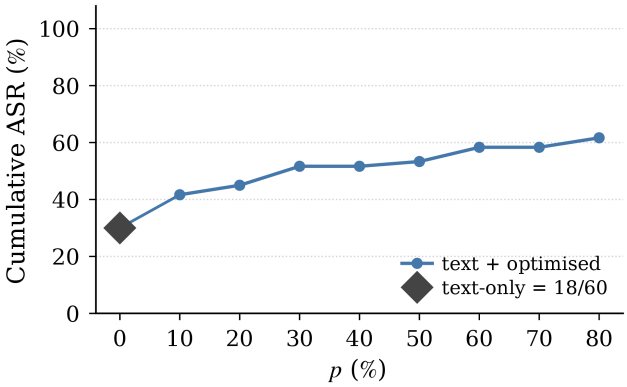

Figure 7. $K{=}1$ Qwen-only cascade against Qwen3-VL-4B-Instruct, as Figure 3 but gated by WildGuard (Table 17, right). Text-only baseline drops to $18/60 = 30\%$; the cascade reaches $37/60 = 61.7\%$ at $p{=}80\%$.

```
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:
```

Table 18. $K{=}4$ cumulative success counts under Qwen3Guard gating (full version of Table 6; visualized in Figure 5). $n{=}60$.

p (%)	Qwen3-VL	Phi-4	Sonnet	GPT-5.4n	Haiku
0 (text)	42	37	37	36	26
10	45	41	39	43	35
20	48	43	44	48	43
30	51	46	49	51	48
40	54	47	52	56	50
50	56	51	53	57	52
60	57	52	54	58	53
70	57	52	54	59	53
80	**57**	**52**	**54**	**59**	**53**

Table 19. $K{=}4$ cumulative success counts under WildGuard gating (visualized in Figure 6). $n{=}60$.

p (%)	Qwen3-VL	Phi-4	Sonnet	GPT-5.4n	Haiku
0 (text)	18	17	15	15	10
10	23	22	18	22	17
20	24	23	20	23	21
30	28	28	25	31	25
40	30	30	27	33	27
50	33	34	29	37	30
60	35	36	30	39	31
70	35	36	30	39	31
80	**38**	**38**	**32**	**41**	**32**

```
{response}

--

JSON evaluation:
```

The judge returns a JSON object whose `attack_success` boolean is the per-row label aggregated into all ASR percentages reported in the paper.

G. Implementation Details

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 $42 + \text{row_idx}$. Candidate batches use a separate random-number-generator stream initialized with the same per-row seed. All target inferences use temperature 0; vLLM text inference sets deterministic PyTorch/cuBLAS options, disables CUDA graphs with eager execution, and uses one sequence per batch.

Table 20. Suffix-length sweep on the 60-prompt cohort against Qwen3-VL-4B-Instruct. Qwen-only suffix-isolated image-anchor objective, matched to the K=1 in-place setting in optimizer budget and embedder. Text-only baseline ASR is 98.3%.

L (tokens)	random_suffix	optimized	Δ (opt $-$ rand)
10	66.7	90.0	$+23.3$
20	70.0	88.3	$+18.3$
30	71.7	**93.3**	$+21.7$
40	83.3	**93.3**	$+10.0$
50	73.3	88.3	$+15.0$
60	75.0	88.3	$+13.3$
70	73.3	86.7	$+13.3$
80	75.0	88.3	$+13.3$

Table 21. Suffix-length cascade against Qwen3-VL-4B-Instruct, gated by Qwen3Guard. Cumulative success counts out of n=60; row L=0 is the text-only baseline. Random-suffix is flat at 42 across every L.

L (tokens)	random-suffix	optimized
0 (text)	42 (70.0%)	42 (70.0%)
10	42 (70.0%)	45 (75.0%)
20	42 (70.0%)	47 (78.3%)
30	42 (70.0%)	47 (78.3%)
40	42 (70.0%)	48 (80.0%)
50	42 (70.0%)	48 (80.0%)
60	42 (70.0%)	48 (80.0%)
70	42 (70.0%)	49 (81.7%)
80	**42 (70.0%)**	**49 (81.7%)**

Hyperparameters. 60-prompt cohort, K=4 ensemble (Section 5.4): N=200, T=256, B=1024, patience π=40, tolerance $\delta = 10^{-4}$, mask_init = placeholder. 60-prompt cohort, K=1 Qwen-only in-place mask (Section 5.2; Table 3): N=100, T=64, B=256, patience π=20, tolerance $\delta = 10^{-4}$, full-prompt scope, mask_init = placeholder. Suffix appendix (Section E): same N, T, B, π, δ as the K=1 in-place run, but with a suffix-isolated Qwen-only image-anchor objective over appended L-token suffixes and no mask_init.

Compute. GCG runs in bf16 on a single GPU ($\sim$6.5 GB for all four embedders). The full sweep parallelizes across cohort rows on 4 H200 GPUs, with each shard binding to its own GPU and processing rows where row_idx $\bmod$ num_shards = rank.