

# Improving GCG: Soft-GCG and Activation-Based Objectives for Adversarial Suffix Optimization

CS2881: AI Alignment and Safety - Fall 2025

Kayden Kehe

Harvard University

kaydenkehe@college.harvard.edu

Ege Cakar

Harvard University

ecakar@college.harvard.edu

Hannah Guan

Harvard University

hguan@college.harvard.edu

## Abstract

The Greedy Coordinate Gradient (GCG) attack demonstrates that aligned language models remain vulnerable to adversarial suffixes, but its discrete optimization is computationally expensive and targets output probabilities rather than internal safety mechanisms. We investigate two approaches to improve GCG along orthogonal axes. First, we propose Activation-Guided GCG, which replaces the standard log-likelihood objective with losses that directly minimize projections onto refusal directions in the model’s residual stream. We find that targeting a single layer-position pair achieves better sample efficiency than baseline GCG, suggesting that representationally-aware objectives may offer advantages over purely behavioral targets. Second, we introduce Soft-GCG, a continuous relaxation using Gumbel-Softmax that achieves 43x speedup over standard GCG while maintaining attack success rate parity. We evaluate Soft-GCG on the Gemma 3 model family, finding that smaller models (1B–4B parameters) remain vulnerable while larger models (12B+) resist the attack, highlighting a capability-robustness correlation in current safety training. Our results suggest that efficient, mechanistically-informed attacks are feasible and that defenses should account for both the speed and internal targeting of adversarial optimization.

## Introduction

The widespread deployment of Large Language Models (LLMs) has necessitated rigorous safety alignment techniques to prevent the generation of harmful or unethical content. At least in public chat interfaces, it seems like these alignment techniques have come very far. However, recent research has demonstrated that these safety guardrails are surprisingly brittle. Two distinct lines of inquiry have emerged to analyze this fragility: automated adversarial attacks on the input space, and the analysis of refusal representations within the model’s residual streams.

The Greedy Coordinate Gradient (GCG) method, proposed by Zou et al. [1], demonstrates that optimizing adversarial suffixes can reliably bypass alignment. However, GCG operates on a behavioral proxy: it optimizes inputs to maximize the likelihood of specific target output tokens (e.g., forcing the model to output “Sure, here is...”). While effective, this objective targets the *symptom* of the jailbreak (the output logits) rather than the *mechanism* of the refusal. Additionally, the GCG method is limited by the given initialization

---

<sup>0</sup>Code: [github.com/Ege-Cakar/ImprovingGCG](https://github.com/Ege-Cakar/ImprovingGCG)

conditions and long waits until convergence. Its discrete optimization methods are much slower compared to continuous methods, and we hypothesize that a large portion of the optimization that is done by GCG can be done first via continuous optimization, and that discrete optimization is only necessary to overcome the “projection gap”, i.e the loss in performance that comes from jumping to discrete tokens at the end after optimizing continuously and achieving a minimum in between real tokens that doesn’t generalize to hard tokens.

Conversely, Ardit et al. [2] investigates the model’s internal representations of refusal. They demonstrated that refusal behavior in open-source chat models is mediated by a single, one-dimensional subspace in the model’s activation space and that simply ablating this direction eliminates refusal behavior. This approach, though, is limited by the need for possible attacks to have white-box access to a model’s activations. Thus, in its purest form, it cannot be used on closed models, which are typically the most widely used platforms for the general public to access language models.

This project is motivated by the hypothesis that GCG has multiple easy avenues that can be explored for improvement. Specifically, we aim to design adversarial suffix attacks which are optimized to elicit the model’s internal refusal direction, as well as attacks that are more efficient than baseline GCG. Our work is done along two main directions:

- **Eliciting refusal behavior via prompt suffixes.** One way we believe GCG can be improved is via *explicitly* targeting a model’s internal representation of refusal, thinking it may offer a more robust or efficient optimization objective. By modifying the GCG algorithm to minimize the projection of the residual stream onto the refusal direction identified by Ardit et al. [2], rather than maximizing the log-likelihood of a target string, we aim to bridge the gap between representation engineering and adversarial optimization. This approach asks a fundamental question: Can we construct white-box attacks that are “representationally aware,” leveraging the internal geometry of safety representations to break alignment more effectively and efficiently?
- **Designing efficient GCG methods.** We also believe that GCG can be made much more efficient by improving the quality of its initialization. We investigate whether a hybrid optimization strategy – utilizing temperature-annealed continuous relaxation for global search used to initialize GCG close to the solution, followed by a minimal discrete refinement due to the “projection gap” problem – can achieve feature parity with GCG. With this, we ask: Is the computational bottleneck introduced by the discrete optimization necessary for the success of GCG?

## Related Work

### Adversarial Attacks on Aligned LLMs

The vulnerability of aligned LLMs to adversarial inputs has been extensively documented. Early “jailbreaks” relied on manual prompt engineering (e.g., “DAN” [3] or role-play scenarios). However, Zou et al. [1] introduced a formal, automated gradient-based approach known as Greedy Coordinate Gradient (GCG). GCG overcomes the discrete nature of text optimization by using token-level gradients to identify promising candidate replacements for an adversarial suffix.

As part of their formulation, they define the adversarial loss  $\mathcal{L}$  for a prompt  $x_{1:n}$  as the negative log probability of a target sequence  $x_{n+1:n+H}^*$ :

$$\mathcal{L}(x_{1:n}) = -\log p(x_{n+1:n+H}^* | x_{1:n}). \quad (1)$$

For each token  $x_i$  in the adversarial suffix, they compute the gradient of the loss with respect to the one-hot encoding  $e_{x_i}$ :

$$\nabla_{e_{x_i}} \mathcal{L}(x_{1:n}) \in \mathbb{R}^{|V|}. \quad (2)$$

They then select the top- $k$  candidates with the largest negative gradient values to evaluate in a forward pass. They demonstrated that this method achieves high Attack Success Rates (ASR), reaching 100% on Vicuna-7B and 88% on Llama-2-7B-Chat for harmful behaviors. Furthermore, they showed that these attacks are transferable, meaning a suffix optimized on one model can often break completely different black-box models. While highly effective, GCG is computationally expensive and relies on the assumption that forcing the first few tokens of output determines the subsequent generation trajectory. In this project, we aim to create a more representative suffix to bypass refusals as well as use continuous optimization to lower the computation overhead.

## Refusal Directions

Parallel to adversarial attacks, researchers have sought to understand how models represent refusal internally using techniques often associated with representation engineering. Ardit et al. [2] utilized a difference-in-means methodology to isolate a single direction  $\hat{r}$  (the normalized difference vector) in the residual stream that differentiates between harmful and harmless instructions. Concretely, they define:

$$r^{(l)} = \mu_{\text{harmful}}^{(l)} - \mu_{\text{harmless}}^{(l)} \quad (3)$$

where  $\mu^{(l)}$  represents the mean activation at layer  $l$ .

They validated this finding through a technique they term **Directional Ablation**. Directional ablation erases the refusal direction from the residual stream activation  $x$  by projecting it out:

$$x' \leftarrow x - (x \cdot \hat{r})\hat{r}. \quad (4)$$

By performing this intervention at every layer (or equivalently, orthogonalizing model weights), they effectively disabled refusal. Crucially, Ardit et al. [2] compared their weight orthogonalization method directly against GCG on the HarmBench benchmark. They found their method achieved comparable or superior Attack Success Rates (ASR) while requiring significantly less computational cost as it does not require per-prompt optimization. Conversely, they demonstrated that adding this direction to harmless prompts ( $x' \leftarrow x + \alpha\hat{r}$ ) induced refusal behavior even on benign queries like “List the benefits of yoga” [2]. Their work suggests that safety fine-tuning does not eliminate harmful knowledge but rather learns a specific “refusal feature” that overrides generation when activated. Our results confirm this conjecture and additionally add flexibility to their activation-based approach by developing a method to generate adversarial suffixes that optimize for bypassing refusal directions.

## Continuous Prompt Optimization

Wen et al. [4] proposed “Hard Prompts Made Easy” (PEZ), a gradient-based framework designed to bridge the optimization gap between continuous embedding spaces and discrete token sequences. Their work addresses the largest limitation in soft prompt tuning: while continuous soft prompts are easier to optimize via gradient descent, they often fail to map onto coherent or effective discrete tokens when simply projected to the nearest vocabulary neighbors after training – a phenomenon the authors describe as the “projection gap.”

To overcome this, PEZ introduces a methodology where the discretization step is integrated directly into the forward pass of the optimization loop. The algorithm maintains a set of learnable continuous parameters (soft

prompts) but projects them onto their nearest discrete neighbors in the embedding matrix before computing the loss. The gradients derived from this discrete forward pass are then used to update the underlying continuous parameters. This effectively allows the optimizer to explore the smooth, differentiable continuous space while being constantly tethered to the valid discrete manifold, preventing the optimization trajectory from drifting into regions of the latent space that have no semantic validity. Empirically, Wen et al. demonstrated that this approach could robustly discover effective hard prompts for both text-to-image generation (inverting CLIP encoders for Stable Diffusion) and text-only tasks. However, in our testing, their approach remained nonperformant for jailbreak purposes.

## Efficient Jailbreak Methods

Several recent approaches have sought to reduce the computational cost of automated jailbreaks. PAIR [5] uses an attacker LLM to iteratively refine adversarial prompts based on target model responses, requiring only black-box access. TAP [6] extends this with tree-of-thought reasoning and pruning for more efficient exploration. AutoDAN [7] employs a hierarchical genetic algorithm to evolve semantically meaningful jailbreak prompts, achieving interpretability that gradient-based methods lack. These approaches trade off gradient information for query efficiency and black-box applicability. Our Soft-GCG occupies a different point in this design space: it retains white-box gradient access but replaces discrete search with continuous optimization, achieving speedups complementary to these query-based methods.

## Methodology

### Activation-Guided GCG

This approach combines the gradient-based optimization framework of GCG with mechanistic insights from refusal direction analysis. We first extract refusal directions following [2], then use these directions to guide adversarial suffix optimization through novel activation-based objectives in the hope that we can improve performance over baseline GCG.

The activation-based approach offers two potential advantages. First, by targeting the causal mechanism underlying refusal (the internal direction identified through contrastive analysis), we may discover more fundamental adversarial perturbations than those found by optimizing surface-level output probabilities. Second, since the refusal direction is computed across many diverse examples, the activation objective may generalize better to unseen harmful prompts compared to token-level objectives that risk overfitting to specific target strings.

### Refusal Direction Extraction

Following [2], we extract refusal directions using the difference-in-means methodology (Equation 3). For each layer  $\ell$ , we compute candidate directions and select the optimal layer  $\ell^*$  by maximizing ablation effectiveness. The selected direction  $\mathbf{d}_{\text{refusal}} = r^{(\ell^*)}$  serves as the optimization target for our attack.

### Activation-Based Adversarial Optimization

Standard GCG [1] optimizes an adversarial suffix by using one-hot relaxation to compute token gradients (Equation 2). The gradients are aggregated across multiple prompts, and the top- $K$  tokens with largest negative gradients are used to generate candidate suffixes via greedy coordinate descent. We adopt this same optimization framework, replacing the optimization objective.

## Activation-Based Objective Functions

The core contribution of this work is replacing the output-based loss (Equation 1) with objectives that directly target internal refusal representations. Given the refusal direction  $\mathbf{d}_{\text{refusal}}$  at layer  $\ell^*$  and position  $p$  (typically the last instruction token), we explore five activation-based objectives that vary in scope and optimization target:

- **Single:** Minimize projections at the most influential layer and position
- **Negative:** Push the projections at the most influential layer and position to be negative (actively opposing refusal)
- **Layer:** Minimize projections across all token positions at the most influential layer
- **Token:** Minimize projections at a single token position across all layers
- **All:** Minimize projections across all layers and all positions

These objectives represent different design choices in the locality-globality spectrum: from targeting a single activation, to suppressing refusal across the entire forward pass. Formulations on each objective can be found in the appendix. We evaluate each independently to assess which strategy most effectively suppresses refusal.

## Soft-GCG

Soft-GCG is our method of relaxing GCG into a continuous optimization problem, developed with the aim of achieving the continuous problem while diverging as little as possible from GCG. Instead of searching for a sequence of discrete tokens (like standard GCG which swaps tokens), Soft-GCG maintains a set of continuous variables (logits) for each position in the suffix. These logits represent a probability distribution over the entire vocabulary.

### Problem Formulation

Let  $x$  denote the user prompt (e.g., "Write a tutorial on how to make a bomb") and  $y$  denote the target response (e.g., "Sure, here is a tutorial..."). We seek an adversarial suffix  $s$  of length  $L$ , consisting of tokens from a vocabulary  $V$ , such that the probability of the target response is maximized:

$$s^* = \arg \max_{s \in V^L} \log P(y | x, s) \quad (5)$$

Since the domain  $V^L$  is discrete, gradients cannot be backpropagated through the token selection indices. Soft-GCG circumvents this by optimizing a continuous parameterization of the suffix.

### Continuous Relaxation via Gumbel-Softmax

Instead of selecting a single discrete token  $s_i$  at each position  $i \in \{1, \dots, L\}$ , we maintain a set of unbounded learnable logits  $\phi_i \in \mathbb{R}^{|V|}$ . These logits represent the unnormalized log-probabilities of selecting each token in the vocabulary. To approximate the discrete selection process while maintaining differentiability, we utilize the Gumbel-Softmax distribution [8]. For each position  $i$ , a "soft" token vector  $\tilde{s}_i$  is computed as:

$$\tilde{s}_i = \text{Softmax} \left( \frac{\phi_i + g_i}{\tau} \right) \quad (6)$$

where:

- $g_i \sim \text{Gumbel}(0, 1)$  are i.i.d. samples drawn from the Gumbel distribution, included to make sure we do not get stuck in local minima in a jagged landscape.
- $\tau > 0$  is a temperature hyperparameter controlling the spikiness of the distribution, starting smoother and allowing more cross-token interaction then pushing the vector into a one hot as it decreases.

The resulting vector  $\tilde{s}_i \in \Delta^{|V|-1}$  lies on the probability simplex. The input embedding for the  $i$ -th suffix position,  $e(s_i)$ , is then approximated by the probability-weighted sum of the vocabulary embeddings matrix  $E \in \mathbb{R}^{|V| \times d}$ :

$$e(\tilde{s}_i) = \tilde{s}_i^\top E = \sum_{v \in V} [\tilde{s}_i]_v \cdot E_v \quad (7)$$

This operation is fully differentiable, allowing gradients from the loss function to flow back to the logits  $\phi$ .

## Optimization Objectives

The primary objective is to minimize the negative log-likelihood of the target sequence  $y$ , conditioned on the prompt  $x$  and the soft suffix  $\tilde{s}$ :

$$\mathcal{L}_{CE}(\phi) = - \sum_{t=1}^{|y|} \log P(y_t | x, \tilde{s}, y_{<t}; \theta) \quad (8)$$

where  $\theta$  represents the fixed parameters of the language model. This directly aligns with the adversarial goal of maximizing target probability.

## Temperature Annealing Schedules

The temperature  $\tau$  plays a critical role in the optimization trajectory. We investigate two annealing schedules:

1. **Linear Schedule.** A simple decay that steadily reduces entropy over  $T$  steps:

$$\tau_t = \tau_{\text{start}} - \left( \frac{t}{T} \right) (\tau_{\text{start}} - \tau_{\text{end}}) \quad (9)$$

Typically annealing from  $\tau = 2.0$  to  $\tau = 0.1$ . This provides a consistent pressure towards discretization.

2. **3-Phase “Slushy” Schedule.** A piecewise schedule designed to balance exploration and refinement:

- **Exploration (0% - 50%)**: High temperature decay ( $2.5 \rightarrow 1.0$ ). Allows broad traversal of the embedding space.
- **Refinement (50% - 75%)**: Moderate temperature ( $1.0 \rightarrow 0.5$ ). The “slushy” phase where structure forms but the distribution remains flexible enough to shift.
- **Solidification (75% - 100%)**: Rapid cooling ( $0.5 \rightarrow 0.01$ ). Forces the soft vectors to snap to the nearest discrete tokens.

## Discretization

After the optimization concludes, the continuous logits  $\phi$  must be projected back to the discrete vocabulary to form the final usable suffix. We select the token with the highest logit value at each position:

$$s_i^{\text{final}} = \arg \max_{v \in V} [\phi_i]_v \quad (10)$$

This discrete suffix is then evaluated against the model to verify its effectiveness. The hope is that the discretization gap is very low at this step, thanks to the latter portions of the optimization being done with an ever-decreasing temperature. From this point on, the converged substring can directly be utilized for attacks or it can be used as the initialization of GCG.

## Experimental Results

### Activation-Guided GCG

#### Setup

All experiments use a single open-weight chat model: Llama-2-7b-chat-hf [9]. For optimization and evaluation we rely on the datasets packaged with `refusal_direction` [2] and `llm_attacks` [1]:

- **Harmful prompts (train):** the `harmful_train` split from the `refusal-direction` dataset, using only the instruction text. These are prompts that explicitly request harmful behavior (e.g., how to cause physical, cyber, or social harm).
- **Harmful prompts (test):** the `harmful_test` split from the same dataset, used only for evaluation.
- **AdvBench behaviors (for GCG baseline):** to train the GCG baseline, we use AdvBench, which contains paired (goal, target) strings describing harmful behaviors and their desiderata.

We use the standard parameters provided in the `llm_attacks` repository for all tests. Every test is trained with 200 discrete optimization steps.

#### Evaluation and Attack Success Metrics

After training a universal suffix for each method, we evaluate it on held-out prompts from the `refusal-direction` dataset.

For a given method and evaluation split, we generate completions by appending the learned suffix to the instruction (for suffix-based methods), or by applying only the internal activation intervention (for ablation baselines). We then score each completion with three complementary methods, each inducing a notion of “harmfulness”.

The following are our evaluation definitions for harmfulness:

- **Substring ASR:** Labels a completion as harmful if it does *not* contain any of a fixed list of refusal-style prefixes (e.g., “I’m sorry”, “I cannot”).
- **LlamaGuard2 ASR:** Labels a completion as harmful if it is labeled unsafe by the LlamaGuard2 safety classifier [10].
- **HarmBench ASR:** Labels a completion as harmful if it is labeled unsafe by the HarmBench evaluation framework [11].

#### Results

**Activation-based objectives show improved attack success despite limited optimization steps. (Table 1)**  
As expected, the baseline (no suffix) results have the lowest ASR scores, and the ablation (steering the activation values) results are the highest. All optimization objectives meaningfully improve the substring

ASR. While absolute ASR remains low across all suffix-based methods, suggesting that 200 optimization steps is insufficient for convergence, the relative improvement of using the “Single” objective over baseline GCG indicates that activation-based objectives may offer better sample efficiency.

Method	ASR (Harmful)		
	Substring	LlamaGuard2	HarmBench
Baseline	0.39	0.00	0.00
Ablation	0.98	0.50	0.56
Single	<b>0.82</b>	<b>0.04</b>	<b>0.06</b>
Negative	<b>0.82</b>	0.01	0.01
All	0.75	0.00	0.00
Layer	0.71	0.00	0.00
Token	0.74	0.00	0.01
GCG	0.76	0.00	0.01

Table 1: Harmful prompt ASR for suffix-based attacks and refusal-direction ablation on the refusal-direction `harmful-test` split. We report results using the suffix with the lowest loss during training.

To determine how well these methods align with the targeted refusal direction, we computed the Frobenius distance and cosine similarity between activations from suffixed prompts and those from ablated, suffix-free prompts.

Method	Cosine Sim.	Frobenius Dist.
Baseline	0.9984	$\approx 3.6 \times 10^2$
GCG	1.0011	$\approx 4.3 \times 10^2$
Single	1.0002	$\approx 3.9 \times 10^2$
Negative	1.0000	$\approx 3.8 \times 10^2$
All	1.0010	$\approx 4.3 \times 10^2$
Layer	1.0013	$\approx 3.9 \times 10^2$
Token	1.0011	$\approx 4.3 \times 10^2$

Table 2: Cosine similarity and Frobenius distance between full suffix-generated activations and their refusal-direction ablated counterparts.

The Frobenius distance results emphasize the fact that the ablations performed in the refusal direction paper are incredibly small, and during the optimization process, we end up inducing much larger changes in the activation space. The cosine similarity results are relatively uninformative for downstream analysis, perhaps as a result of the curse of dimensionality.

## Soft-GCG

**SGCG by itself achieves performance parity with GCG in our experiments, showing no need for any pure GCG steps at the end. (Figure 1)** Due to the long time it takes to run GCG, we generate one subscript for each configuration for our long-main run. Then the results are evaluated against the entire AdvBench corpus multiple times to accurately justify results. However, this performance parity is also experienced in

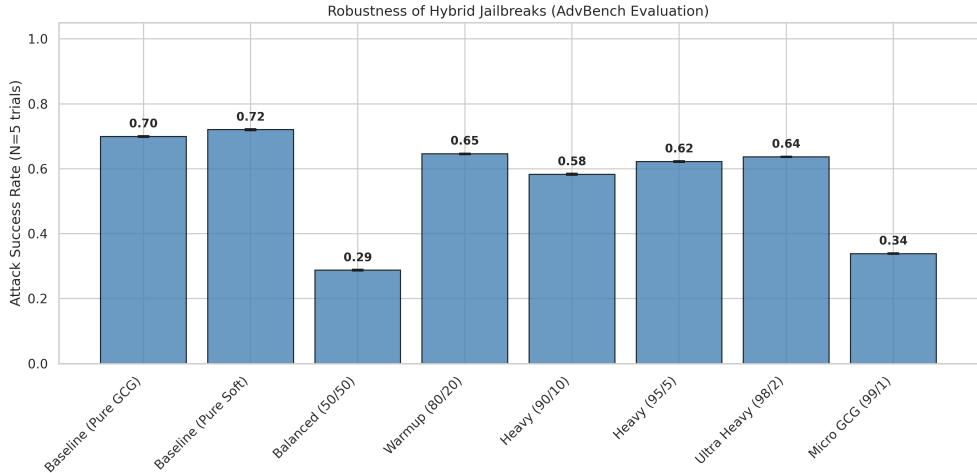


Figure 1: Performance of different configurations of SGCG→GCG. We see that it is more useful to run more SGCG iterations than GCG (number of iterations are constant).

our smaller-scale runs with SGCG vs GCG. The results are evaluated on all prompts from AdvBench using Llama-2-7b-chat-hf and with substring ASR, with the same substrings from the original GCG paper. For the sweep experiments, we used a split of 25 training prompts and 10 test prompts, all pulled from AdvBench.

## Hyperparameters

The Suffix Length is fixed at  $L = 20$  tokens. For the optimization steps in the sweep experiments, we varied combinations of Soft and GCG steps, including Pure GCG (500 steps), Pure Soft (500 steps), and Hybrid configurations (e.g., 250/250, 400/100, etc.). For the Gemma experiments, we ran 500 steps of Pure Soft optimization after the sweep results. The Learning Rate is 0.1 for the Adam optimizer applied to suffix logits, and **the batch size** is 10 for Gemma experiments; with varying batch sizes for GCG candidates (128) in the sweep.

The main benefit of SGCG over GCG is the speed increase: **SGCG is 43 times faster than GCG**, finishing at roughly 1 minute and 54 seconds, as opposed to 81 minutes and 20 seconds. This is on the same hardware, wall-clock time, with the exact same number of iterations (500).

We then employ SGCG to attack the Gemma 3 family of models to test the strength of frontier models against this attack. (Table 3) We chose to evaluate Gemma 3 with our SGCG because a) Ollama, one of the most popular local serving apps, claims it is the best model family that can run on a single GPU [12] and b) it is from a large AI lab that people know about (Google) and would be inclined to download [13]. As such, we believed Gemma 3 to be the most realistic local model to deploy this on.

## Discussion and Future Work

### Theory of Change

We develop two complementary contributions exposing critical vulnerabilities in aligned language models: (1) a novel white-box attack targeting internal refusal representations, demonstrating that safety mechanisms encoded as linear directions can be systematically suppressed via adversarial optimization, and (2) a 43x speed improvement to GCG that reduces computational barriers dramatically. By achieving high attack

<b>Model</b>	<b>Substring ASR*</b>
Gemma3-270m	$1.000 \pm 0.000$
Gemma3-1b	$0.577 \pm 0.031$
Gemma3-4b	$0.336 \pm 0.104$
Gemma3-12b	$0.000 \pm 0.000$

Table 3: ASR of the Gemma3 family. The 27b model was not evaluated due to the performance of the 12b model. ASR is marked with an asterisk because the 270m model became incoherent after optimization.

success rates on modern models (up to 34% ASR on Gemma 3 4B), we provide concrete evidence that current RLHF-based alignment is insufficient against representation-level attacks. The speed improvement transforms attacks from resource-intensive operations into ones executable within minutes on consumer hardware, making this a practical exploit accessible to broader actors. We demonstrate success on frontier models and call for improved safety training as small language models become more capable.

Our theory relies on the security mindset: publicly disclosing attack methodologies – including efficiency improvements – accelerates defense development faster than exploitation. We assume (1) white-box access requirements limit immediate harm despite increased speed, (2) currently exploitable models aren’t yet smart enough to be destructive, (3) AI labs will prioritize robustness when vulnerabilities are demonstrated to be effective and rapidly exploitable, and (4) insights from current models remain relevant for future architectures. The 43x speedup demonstrates that computational protections are shallow, compelling faster defensive action. The ultimate impact is next-generation AI systems developed with representation-level robustness: safety mechanisms distributed beyond exploitable linear directions, adversarial training including efficient activation-based attacks, and red-teaming testing for mechanistic vulnerabilities discoverable at scale. The primary risk is that findings improve attacks rather than defenses before adequate countermeasures are deployed.

## Limitations

Even though we argue that Gemma 3 is probably the best model to test this on, it does come with its problems. The most significant one is the ginormous vocabulary size of 262 thousand for all Gemma 3 models[13], as opposed to something like 128 thousand for Llama 3 [14]. Due to the way we parametrize SGCG, the vocabulary size directly increases the complexity of the problem, making it a harder optimization. This, however, is entangled with the safety training of specific models, so we were unable to isolate the effect of vocabulary size directly on performance.

There are also limitations in how we evaluate our methods. Particularly, we use substring matching to calculate ASR for both Activation-Guided GCG and Soft-GCG. Substring matching can lead to both false negatives and false positives, since it isn’t adaptable to the response itself and solely looks for common strings like “I cannot”, etc to determine refusal. Future work could involve using more robust metrics of evaluation, such as using LlamaGuard as a judge.

## References

- [1] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.
- [2] Andy Ardit, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *Advances in Neural Information Processing Systems*, volume 38, 2024.
- [3] Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. “do anything now”: Characterizing and evaluating in-the-wild jailbreak prompts on large language models. *arXiv preprint arXiv:2308.03825*, 2023.
- [4] Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery, 2023.
- [5] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries, 2023.
- [6] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box llms automatically, 2023.
- [7] Xiaogeng Liu, Nan Xu, Muhan Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. In *The Twelfth International Conference on Learning Representations*, 2024.
- [8] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax, 2017.
- [9] Hugo Touvron et al. Llama 2: Open foundation and fine-tuned chat models, 2023.
- [10] Llama Team. Meta llama guard 2. [https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL\\_CARD.md](https://github.com/meta-llama/PurpleLlama/blob/main/Llama-Guard2/MODEL_CARD.md), 2024.
- [11] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaei, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal, 2024.
- [12] Ollama – gemma 3. <https://ollama.com/library/gemma3>.
- [13] Google Developers Blog. Gemma explained: What’s new in gemma 3. <https://developers.googleblog.com/en/gemma-explained-whats-new-in-gemma-3/>, 2025.
- [14] Hugging Face. Welcome llama 3 – meta’s new open llm. <https://huggingface.co/blog/llama3>, 2024.

## Appendix

### Activation-Guided GCG Objective Functions

**Single.** Minimize projections at the most influential layer and position

$$\mathcal{L}_{\text{act}}^{\text{zero}}(\mathbf{x}) = \left( \frac{\mathbf{h}_p^{(\ell^*)}(\mathbf{x}) \cdot \mathbf{d}_{\text{refusal}}}{\|\mathbf{d}_{\text{refusal}}\|} \right)^2 \quad (11)$$

where  $\mathbf{h}_p^{(\ell^*)}(\mathbf{x})$  is the hidden state at layer  $\ell^*$  and position  $p$  for input  $\mathbf{x} = \mathbf{x}_{\text{inst}} \oplus \mathbf{x}_{\text{adv}}$ .

**Negative Projection.** Push the projection to be negative (actively opposing refusal):

$$\mathcal{L}_{\text{act}}^{\text{neg}}(\mathbf{x}) = \frac{\mathbf{h}_p^{(\ell^*)}(\mathbf{x}) \cdot \mathbf{d}_{\text{refusal}}}{\|\mathbf{d}_{\text{refusal}}\|}. \quad (12)$$

**Layer.** Minimize projections across all token positions at the most influential layer:

$$\mathcal{L}_{\text{act}}^{\text{layer}}(\mathbf{x}) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \left( \frac{\mathbf{h}_p^{(\ell^*)}(\mathbf{x}) \cdot \mathbf{d}_{\text{refusal}}}{\|\mathbf{d}_{\text{refusal}}\|} \right)^2 \quad (13)$$

where  $\mathcal{P}$  represents all token positions in the sequence.

**Token.** Minimize projections at a single token position across all layers:

$$\mathcal{L}_{\text{act}}^{\text{token}}(\mathbf{x}) = \frac{1}{L} \sum_{\ell=1}^L \left( \frac{\mathbf{h}_p^{(\ell)}(\mathbf{x}) \cdot \mathbf{d}_{\text{refusal}}^{(\ell)}}{\|\mathbf{d}_{\text{refusal}}^{(\ell)}\|} \right)^2 \quad (14)$$

where  $L$  is the total number of layers and  $\mathbf{d}_{\text{refusal}}^{(\ell)}$  is the refusal direction at layer  $\ell$ .

**All.** Minimize projections across all layers and all positions:

$$\mathcal{L}_{\text{act}}^{\text{global}}(\mathbf{x}) = \frac{1}{L \cdot |\mathcal{P}|} \sum_{\ell=1}^L \sum_{p \in \mathcal{P}} \left( \frac{\mathbf{h}_p^{(\ell)}(\mathbf{x}) \cdot \mathbf{d}_{\text{refusal}}^{(\ell)}}{\|\mathbf{d}_{\text{refusal}}^{(\ell)}\|} \right)^2. \quad (15)$$