



ÉCOLE POLYTECHNIQUE FÉDÉRALE DE  
LAUSANNE

MASTER RESEARCH PROJECT

# Multimodal Adversarial Attacks for Vision–Language Models

**Student:** Daniel López Gala

**Supervisor:** Elías Abad Rocamora

**Professor:** Volkan Cevher

**Laboratory:** LIONS – Laboratory for Information and Inference  
Systems

June 6, 2025

# Contents

<b>1</b>	<b>Abstract</b>	<b>1</b>
<b>2</b>	<b>Introduction</b>	<b>1</b>
2.1	Vision–Language Models . . . . .	1
2.2	Adversarial Vulnerabilities . . . . .	1
2.3	Joint Multimodal Attack Framework . . . . .	1
2.4	Applications and Evaluation . . . . .	2
2.5	Contributions and Organization . . . . .	2
<b>3</b>	<b>Related work</b>	<b>2</b>
3.1	Vision-only adversarial attacks . . . . .	2
3.2	Text-only adversarial attacks . . . . .	2
3.3	Multimodal adversarial attacks . . . . .	3
3.4	Defenses for VLMs . . . . .	3
<b>4</b>	<b>Methods</b>	<b>3</b>
4.1	Text-Only Adversarial Suffix Optimization via GCG . . . . .	3
4.2	Overall Optimization Problem . . . . .	4
4.3	Attack Algorithm . . . . .	5
4.3.1	Key Definitions . . . . .	5
4.3.2	Attack Overview . . . . .	6
4.4	Experimental setup . . . . .	7
4.5	Performance evaluation . . . . .	7
4.6	Dynamic search-width scheduling . . . . .	7
<b>5</b>	<b>Results</b>	<b>9</b>
5.1	Joint Evaluation Results . . . . .	9
5.2	Dynamic Search-Width Results . . . . .	9
<b>6</b>	<b>Discussion</b>	<b>10</b>
	<b>Appendix A: Qualitative Example of a Joint PGD+GCG Attack</b>	<b>14</b>
	<b>Appendix B: Notation Guide</b>	<b>15</b>

# 1 Abstract

Vision-language models (VLMs), such as LLaVA and Gemma, show great multimodal understanding, but are vulnerable to adversarial attacks. In this work, we develop a unified multimodal adversarial attack. This combines image perturbations via Projected Gradient Descent (PGD) and adversarial text suffix optimization using Greedy Coordinate Gradient (GCG). This joint approach optimises both modalities simultaneously and outperforms unimodal strategies in terms of attack efficacy. Evaluations demonstrate a high success rate in causing VLMs to produce unsafe or misaligned outputs, revealing critical vulnerabilities and contributing to the development of defence strategies.

## 2 Introduction

### 2.1 Vision–Language Models

Vision–language models (VLMs) such as LLaVA Touvron et al. [2023] and Gemma-3-4b-it Dettmers et al. [2023] have impressive capabilities in a wide range of multimodal tasks, including image captioning, visual question answering, and dialogue. These systems combine a powerful visual encoder (CLIP-based backbone Radford et al. [2021], Zhai et al.) with a large language model to interpret and generate natural language grounded in visual inputs.

### 2.2 Adversarial Vulnerabilities

However, like their unimodal counterparts, VLMs remain vulnerable to adversarial perturbations in both the vision and language domains, leaving the door open for attackers to exploit them and make the LLM generate malicious outputs, like instructions to make bombs or to hack governmental organizations Zou et al. [2023], Schlarman and Hein. Adversarial examples for images are small, often imperceptible perturbations to pixel values. They have been extensively studied, with PGD as the standard attack and defence benchmark Madry et al. [2018]. Separately, work on adversarial prompt optimisation has shown that aligned language models can be "jailbroken" by appending a carefully optimised sequence of tokens using GCG methods Zou et al. [2023]. However, attacks targeting only one modality leave the other modality intact, potentially limiting overall strength and transferability against VLMs.

### 2.3 Joint Multimodal Attack Framework

In this project, we extend these unimodal techniques to design and evaluate fully joint multimodal attacks against VLMs. Concretely, we integrate PGD for perturbing input images with GCG for optimising adversarial text suffixes in a single forward–backward pass: gradients are computed once for both modalities, the image is updated via PGD, and then all GCG text-edit candidates are evaluated on the updated adversarial image to select the best textual perturbation. By evaluating every text candidate on the same updated image, we capture the full cross-modal interactions between image and text perturbations. This yields dversarial image-text pairs that are significantly stronger than those obtained by separate or alternating optimisation, while still requiring only one

backward pass per iteration (see Algorithm 2). Our unified pipeline supports three modes of operation:

- **PGD-only:** perturb the image while keeping the text prompt fixed;
- **GCG-only:** optimise the textual suffix with the clean image;
- **Joint PGD+GCG:** update both image and text jointly as described above.

## 2.4 Applications and Evaluation

We apply these attacks to VLMs such as LLaVA, LLaVA-RC (LLaVA with Robust CLIP Schlarmann et al. [2024]), and Gemma-3-4b-it, and measure success by whether the perturbed inputs induce unsafe or misaligned outputs under a moderation filter (Llama Guard Inan et al. [2023]). We observe that the Gemma models, particularly Gemma-3-4b-it with GemmaShield2, are more robust; for instance, PGD-only attacks are completely blocked, though GCG-only and joint PGD+GCG attacks can still be effective.

## 2.5 Contributions and Organization

This report is organized as follows. Section 3 surveys existing adversarial methods in vision, language, and multimodal settings. In Section 4 we detail our joint PGD+GCG algorithm, including the dynamic search-width schedule. Section 4.5 describes our evaluation scripts and metrics (e.g. AS@K), and Section 5 presents findings on attack strength and efficiency. Finally, Section 6 discusses promising approaches and future work.

# 3 Related work

## 3.1 Vision-only adversarial attacks

Projected Gradient Descent (PGD) established the standard for white-box image attacks and defenses Madry et al. [2018]. Croce and Hein [2020] later showed that assembling a small ensemble of PGD variants and complementary attacks (AutoAttack) yields reliable, parameter-free robustness evaluations .

## 3.2 Text-only adversarial attacks

Zou et al. [2023] introduced Greedy Coordinate Gradient (GCG), using token-level gradients to craft universal jailbreak suffixes that transfer across aligned LLMs. Jia et al. [2024] improved on GCG—dubbed I-GCG—by diversifying target templates, adaptively selecting multiple token updates per step, and easy-to-hard initialization, achieving near-100% success on standard jailbreak benchmarks. Andriushchenko et al. [2024] subsequently showed that simple adaptive strategies (random log-prob search, transfer attacks, API-specific tricks) can jailbreak even the most safety-aligned LLMs with 100% success.

### 3.3 Multimodal adversarial attacks

Early multimodal attacks focused on the vision branch of VLMs, demonstrating that small image perturbations can mislead visual question answering and grounded dialogue models Carlini and Wagner [2023]. Yin et al. [2023] proposed VLATTACK to fuse single-modal image block-wise similarity attacks with independent text perturbations, then jointly optimize image-text pairs via an iterative cross-search scheme. Wang et al. [2024] went further under a white-box setting, first crafting an adversarial image prefix from noise, then co-optimizing a text suffix to maximize toxic affirmative responses—achieving a 96% success rate on MiniGPT-4.

### 3.4 Defenses for VLMs

On the defense side, Schlarmann et al. [2024] introduced Robust CLIP, an unsupervised adversarial fine-tuning of the CLIP encoder that closes stealth-attack blind spots for downstream VLMs without retraining them.

## 4 Methods

### 4.1 Text-Only Adversarial Suffix Optimization via GCG

Before describing our joint multimodal attack, we first review the Greedy Coordinate Gradient (GCG) method for crafting adversarial text suffixes on aligned LLMs Zou et al. [2023].

**Adversarial Objective.** Given a clean prompt token sequence  $\mathbf{T} = (t_1, \dots, t_n)$  and a target jailbreak continuation  $\mathbf{Y}^* = (y_{n+1}, \dots, y_{n+H})$ , we define an adversarial prompt  $\mathbf{T}_{\text{adv}} = \mathbf{T} \oplus \mathbf{S}$ . Here,  $\mathbf{S} = (s_1, \dots, s_L)$  is an adversarial suffix composed of  $L$  tokens, and  $\oplus$  denotes the concatenation operator. The suffix  $\mathbf{S}$  is optimized to minimize the negative log-likelihood of generating  $\mathbf{Y}^*$ :

$$\min_{\mathbf{S}} \mathcal{L}(\mathbf{T}_{\text{adv}}) = - \sum_{i=1}^H \log p(y_{n+i} \mid \mathbf{T}_{\text{adv}} \oplus (y_{n+1}, \dots, y_{n+i-1})).$$

The optimization is performed over the tokens  $s_j$  of the suffix  $\mathbf{S}$ , where each token is selected from the vocabulary  $V$ . The set of editable positions, denoted  $\mathcal{I}$  in some formulations (i.e.,  $T_{\text{adv},k}$  for  $k \in \mathcal{I}$ ), corresponds to the token positions of  $\mathbf{S}$  within the full adversarial prompt  $\mathbf{T}_{\text{adv}}$ .  $V$  represents the vocabulary size.

**Gradient-guided Token Selection.** Although the adversarial prompt  $\mathbf{T}_{\text{adv}}$  is a sequence of discrete tokens, we can obtain gradients by differentiating through the token embedding layer  $E(\cdot)$ . Let  $L_{\text{adv}}$  be the length of  $\mathbf{T}_{\text{adv}}$ . The matrix of embedding vectors for  $\mathbf{T}_{\text{adv}}$  is  $\mathbf{E}_{\text{adv}} = E(\mathbf{T}_{\text{adv}}) \in \mathbb{R}^{L_{\text{adv}} \times d}$ . The gradient of the scalar loss  $\mathcal{L}$  with respect to these embeddings is the matrix:

$$\mathbf{G}_{\text{emb}} = \nabla_{\mathbf{E}_{\text{adv}}} \mathcal{L}(\mathbf{T}_{\text{adv}}) \in \mathbb{R}^{L_{\text{adv}} \times d}.$$

For each editable position  $j \in \mathcal{I}$  within  $\mathbf{T}_{\text{adv}}$ , let  $\mathbf{g}_{\text{emb},j} \in \mathbb{R}^d$  be the  $j$ -th row of  $\mathbf{G}_{\text{emb}}$  (i.e., the gradient vector for the embedding of the  $j$ -th token). To determine which

vocabulary token would be optimal at this position, this embedding gradient is used to estimate the gradient with respect to one-hot token representations. Let  $\mathbf{W}_E \in \mathbb{R}^{V \times d}$  be the vocabulary embedding matrix (where  $V$  is the vocabulary size). The gradient vector relevant for token selection at position  $j$ ,  $\mathbf{g}_{\text{vocab},j} \in \mathbb{R}^V$ , can be computed as  $\mathbf{g}_{\text{vocab},j} = \mathbf{g}_{\text{emb},j} \mathbf{W}_E^T$ . To find tokens that would most increase the loss, we select candidates from the vocabulary indices that have the largest values in the negative of this gradient vector:

$$X_j = \text{Top-k}(-\mathbf{g}_{\text{vocab},j}) \subseteq \{1, \dots, V\}, \quad |X_j| = k.$$

**Greedy Coordinate Gradient (GCG) Algorithm.** At each of  $N_{\text{text}}$  iterations, GCG samples  $B$  one-token edits from the union of the  $X_j$ , evaluates the true loss  $\mathcal{L}$  for each candidate via a forward pass, and accept the single best edit. Pseudocode appears in Algorithm 1.

---

**Algorithm 1** Greedy Coordinate Gradient (GCG) for Text-Only Jailbreaks

---

**Require:** Original tokens  $\mathbf{T}$ , editable positions  $\mathcal{I}$ , target  $\mathbf{Y}^*$ , vocabulary embedding matrix  $\mathbf{W}_E$ ,  $k$ ,  $B$ ,  $N_{\text{text}}$

```

1:  $\mathbf{T}_{\text{adv}} \leftarrow \mathbf{T}$ 
2: for  $i = 1, \dots, N_{\text{text}}$  do
3:   Let  $\mathbf{E}_{\text{adv}} = E(\mathbf{T}_{\text{adv}})$ 
4:   Compute gradient matrix  $\mathbf{G}_{\text{emb}} = \nabla_{\mathbf{E}_{\text{adv}}} \mathcal{L}(\mathbf{T}_{\text{adv}})$ 
5:   for each  $j \in \mathcal{I}$  do
6:     Let  $\mathbf{g}_{\text{emb},j}$  be the  $j$ -th row of  $\mathbf{G}_{\text{emb}}$ 
7:     Compute  $\mathbf{g}_{\text{vocab},j} \leftarrow \mathbf{g}_{\text{emb},j} \mathbf{W}_E^T$ 
8:      $X_j \leftarrow \text{Top-k}(-\mathbf{g}_{\text{vocab},j})$ 
9:   end for
10:   $\{\tilde{\mathbf{T}}^{(b)}\}_{b=1}^B \leftarrow$  sample  $B$  candidates by choosing random  $(j_b, v_b)$  with  $v_b \in X_{j_b}$ 
11:   $\ell_b \leftarrow \mathcal{L}(\tilde{\mathbf{T}}^{(b)})$  for  $b = 1, \dots, B$ 
12:   $b^* = \arg \min_b \ell_b$ ,  $\mathbf{T}_{\text{adv}} \leftarrow \tilde{\mathbf{T}}^{(b^*)}$ 
13: end for
14: Return  $\mathbf{T}_{\text{adv}}$ 

```

---

## 4.2 Overall Optimization Problem

We target a fixed jailbreak sequence  $\mathbf{Y}^* = (y_{n+1}, \dots, y_{n+H})$ . Our goal is to jointly find an imperceptible image perturbation  $\boldsymbol{\delta}_I$  and an adversarial text suffix  $\mathbf{S}$  that, when applied to a clean image-text pair  $(\mathbf{I}, \mathbf{T})$ , cause the Vision-Language Model (VLM) to generate  $\mathbf{Y}^*$ . The adversarial text prompt is constructed as  $\mathbf{T}_{\text{adv}} = \mathbf{T} \oplus \mathbf{S}$ , where  $\oplus$  denotes token sequence concatenation.

The joint optimization problem is formulated as:

$$\min_{\boldsymbol{\delta}_I, \mathbf{S}} \mathcal{L}(\mathbf{I} + \boldsymbol{\delta}_I, \mathbf{T} \oplus \mathbf{S}), \quad \text{subject to } \|\boldsymbol{\delta}_I\|_{\infty} \leq \epsilon_I.$$

The constraints on the adversarial suffix  $\mathbf{S}$  (such as its length, vocabulary, and specific token choices) are managed by the Greedy Coordinate Gradient (GCG) optimization procedure, as detailed in Section 4.1 and the subsequent discussion on budget enforcement.

The sequence-generation loss  $\mathcal{L}$ , given the perturbed image  $\mathbf{I}' = \mathbf{I} + \delta_I$  and the adversarial prompt  $\mathbf{T}_{\text{adv}} = \mathbf{T} \oplus \mathbf{S}$ , is defined as:

$$\mathcal{L}(\mathbf{I}', \mathbf{T}_{\text{adv}}) = - \sum_{i=1}^H \log p(y_{n+i} \mid \mathbf{T}_{\text{adv}} \oplus (y_{n+1}, \dots, y_{n+i-1}); \mathbf{I}').$$

Here,  $\delta_I \in \mathbb{R}^{H \times W \times C}$  represents the continuous image perturbation, bounded by  $\epsilon_I$  in  $\ell_\infty$ -norm. The adversarial suffix  $\mathbf{S}$  consists of discrete tokens, selected and optimized via the GCG method.

**Budget Enforcement.** For the image perturbation  $\delta_I$  we enforce the  $\ell_\infty$  budget via the usual PGD projection:

$$\delta_I \leftarrow \Pi_{\|\cdot\|_\infty \leq \epsilon_I} [\delta_I + \alpha_I \text{sign}(\mathbf{g}_I)], \quad \Pi_{\|\cdot\|_\infty \leq \epsilon_I}(X)_{p,q} = \min\{\max\{X_{p,q}, -\epsilon_I\}, \epsilon_I\}.$$

On the other hand, the textual perturbation in GCG is *discrete*, so rather than projecting a continuous  $\ell_\infty$  we bound our attack by

- the set of editable positions  $\mathcal{I}$ ,
- the total number of editing steps  $N_{\text{text}}$ , and
- choosing each replacement only from the top- $k$  gradient-guided token candidates  $X_j$  at each step.

This discrete search over a fixed number of token substitutions naturally limits the magnitude of each edit without requiring an explicit  $\ell_\infty$  projection on the textual perturbation (see Algorithm 1).

### 4.3 Attack Algorithm

We integrate PGD on the image and GCG on the text into a single forward-backward pass that captures cross-modal interactions.

#### 4.3.1 Key Definitions

**Gradients:** Let  $\mathbf{I}' = \mathbf{I} + \delta_I$  be the perturbed image and  $\mathbf{T}_{\text{adv}}$  be the current adversarial text. Let  $\mathbf{E}_{\text{adv}} = E(\mathbf{T}_{\text{adv}})$ .

$$\mathbf{g}_I = \nabla_{\mathbf{I}} \mathcal{L}(\mathbf{I}', \mathbf{T}_{\text{adv}}), \quad \mathbf{G}_{\text{emb}} = \nabla_{\mathbf{E}_{\text{adv}}} \mathcal{L}(\mathbf{I}', \mathbf{T}_{\text{adv}}).$$

**Top- $k$  token set  $X_j$ :** For each editable position  $j \in \mathcal{I}$ , let  $\mathbf{g}_{\text{emb},j}$  be the  $j$ -th row of  $\mathbf{G}_{\text{emb}}$ . Compute  $\mathbf{g}_{\text{vocab},j} = \mathbf{g}_{\text{emb},j} \mathbf{W}_E^T$  (requires vocabulary embedding matrix  $\mathbf{W}_E$ ). Then,

$$X_j = \text{Top-k}(-\mathbf{g}_{\text{vocab},j}),$$

the  $k$  vocabulary indices that most decrease the loss at position  $j$ . Note:  $k$  is the *number of token choices per position*, not the total number of candidates.

**Batch size  $B$  (search width):** At each iteration we sample  $B$  one-token edits from the union of  $\{X_j\}$  and evaluate them on the perturbed image.  $B$  controls the search width.

### 4.3.2 Attack Overview

1. **Gradient computation:** Compute image gradient  $\mathbf{g}_I$  and text embedding gradient matrix  $\mathbf{G}_{\text{emb}}$  (as defined in Sec 4.3.1) in one backward pass.

2. **PGD update (image):**

$$\boldsymbol{\delta}_I \leftarrow \Pi_{\|\cdot\|_\infty \leq \epsilon_I} [\boldsymbol{\delta}_I + \alpha_I \text{sign}(\mathbf{g}_I)], \quad \mathbf{I}' = \mathbf{I} + \boldsymbol{\delta}_I.$$

3. **GCG candidate generation (text):**

- For each  $j \in \mathcal{I}$ , form  $X_j = \text{Top-k}(-\mathbf{g}_{\text{vocab},j})$  (derived from  $\mathbf{G}_{\text{emb}}$  as in Sec 4.3.1).
- Sample  $B$  pairs  $(j_b, v_b)$  with  $j_b \in \mathcal{I}$ ,  $v_b \in X_{j_b}$ .
- For each  $b$ , form  $\tilde{\mathbf{T}}^{(b)}$  by replacing position  $j_b$  in  $\mathbf{T}_{\text{adv}}$  with token  $v_b$ .

4. **Joint evaluation:** for each candidate  $b$  evaluate

$$\ell_b = \mathcal{L}(\mathbf{I}', \tilde{\mathbf{T}}^{(b)})$$

(using the same cached image features from  $\mathbf{I}'$ ).

5. **Selection:** pick  $b^* = \arg \min_b \ell_b$  and set  $\mathbf{T}_{\text{adv}} \leftarrow \tilde{\mathbf{T}}^{(b^*)}$ .

---

#### Algorithm 2 Joint PGD + GCG Attack

---

**Require:** Clean image  $\mathbf{I}$ , initial tokens  $\mathbf{T}$ , modifiable positions  $\mathcal{I}$ , target sequence  $\mathbf{Y}^*$

**Require:** PGD budget  $\epsilon_I$ , step size  $\alpha_I$ , GCG top- $k$ , batch size  $B$ , iterations  $N$ , vocabulary embedding matrix  $\mathbf{W}_E$

```

1:  $\boldsymbol{\delta}_I \leftarrow \mathbf{0}$ ,  $\mathbf{T}_{\text{adv}} \leftarrow \mathbf{T}$  ▷ Initialize
2: for  $i = 1, \dots, N$  do
3:   Let  $\mathbf{I}_{\text{pert}} = \mathbf{I} + \boldsymbol{\delta}_I$  and  $\mathbf{E}_{\text{adv}} = E(\mathbf{T}_{\text{adv}})$ 
4:    $\mathbf{g}_I \leftarrow \nabla_{\mathbf{I}} \mathcal{L}(\mathbf{I}_{\text{pert}}, \mathbf{T}_{\text{adv}})$ ,  $\mathbf{G}_{\text{emb}} \leftarrow \nabla_{\mathbf{E}_{\text{adv}}} \mathcal{L}(\mathbf{I}_{\text{pert}}, \mathbf{T}_{\text{adv}})$  ▷ Compute gradients
5:    $\boldsymbol{\delta}_I \leftarrow \Pi_{\|\cdot\|_\infty \leq \epsilon_I} (\boldsymbol{\delta}_I + \alpha_I \text{sign}(\mathbf{g}_I))$ ,  $\mathbf{I}' \leftarrow \mathbf{I} + \boldsymbol{\delta}_I$  ▷ PGD image step
6:    $\mathbf{v}_{\text{feat}} \leftarrow \phi_V(\mathbf{I}')$  ▷ Cache image features
7:
8:   for each  $j \in \mathcal{I}$  do
9:     Let  $\mathbf{g}_{\text{emb},j}$  be the  $j$ -th row of  $\mathbf{G}_{\text{emb}}$ 
10:    Compute  $\mathbf{g}_{\text{vocab},j} \leftarrow \mathbf{g}_{\text{emb},j} \mathbf{W}_E^T$ 
11:     $X_j \leftarrow \text{Top-k}(-\mathbf{g}_{\text{vocab},j})$  ▷ Top- $k$  candidates
12:   end for
13:
14:   for  $b = 1, \dots, B$  do
15:     sample  $j_b \sim \mathcal{I}$ ,  $v_b \sim X_{j_b}$  ▷ Candidate sampling
16:      $\tilde{\mathbf{T}}^{(b)} \leftarrow \mathbf{T}_{\text{adv}}$  ▷ Create candidate text
17:      $\ell_b \leftarrow \mathcal{L}(\mathbf{I}', \tilde{\mathbf{T}}^{(b)})$  ▷ Compute loss
18:   end for
19:    $b^* = \arg \min_b \ell_b$ ,  $\mathbf{T}_{\text{adv}} \leftarrow \tilde{\mathbf{T}}^{(b^*)}$  ▷ Select best candidate and update suffix
20: end for
21: Output:  $(\mathbf{I} + \boldsymbol{\delta}_I, \mathbf{T}_{\text{adv}})$  ▷ Adversarial pair

```

---



## 4.4 Experimental setup

**Environment.** All experiments were run on Red Hat Enterprise Linux 8.5 with CUDA 12.7 and NVIDIA driver 565.57.01 on a multi-GPU A100-SXM4-80 GB cluster (eight GPUs per node). We use a Conda environment created with Python 3.10, PyTorch 2.6.0+cu126 and HuggingFace Transformers 4.50.2.

**Data.** We attack and evaluate on 5 to 20 held-out pairs from the AdvBench harmful behaviours benchmark. For each pair of prompt and target, one attack is performed to obtain an adversarial suffix and image, and then is evaluated as explained in Section 4.5.

## 4.5 Performance evaluation

Each example  $(\mathbf{I}^{(m)}, \mathbf{T}^{(m)})$  is attacked under each mode (PGD only, GCG only, PGD-GCG, joint PGD+GCG) with different parameters, and then evaluated.

**Metrics.** For each example  $m$ , target sequence  $\mathbf{Y}^*$ , and attack run, we generate up to  $K$  continuations per iteration and classify each with a moderation filter, Llama Guard. We record:

- **AS@K:** This is an indicator  $s_{m,K}$  defined as:

$$s_{m,K} = \mathbf{1}\left(\exists k \leq K : \text{mod}(\text{Gen}_k(\mathbf{I}_{\text{adv}}^{(m)}, \mathbf{T}_{\text{adv}}^{(m)})) = \text{"unsafe"}\right),$$

which indicates whether or not there was a successful attack (i.e., an "unsafe" generation) within  $K$  attempts for a given example  $m$ . We then aggregate this across all  $M$  test examples to calculate the overall attack success rate:

$$\text{Success@K} = \frac{1}{M} \sum_{m=1}^M s_{m,K}, \quad K \in \{5, 20, 50, 100\}.$$

- **Time per iteration:** The average computational time taken for a single iteration of the attack algorithm, including its standard deviation.
- **Average loss:** The mean and standard deviation of the attack loss function  $\mathcal{L}(\mathbf{I}_{\text{adv}}, \mathbf{T}_{\text{adv}})$  computed over the final iterations of the attack.

## 4.6 Dynamic search-width scheduling

To reduce the computational cost of evaluating the adversarial loss over  $N$  iterations, we employ a simple linear decay schedule for the GCG candidate batch size (search width)  $B_i$ , starting from a maximum  $B_{\max}$  and tapering down to a minimum floor  $B_{\min}$ :

$$B_i = \max\left(B_{\min}, B_{\max} \left(1 - \frac{i-1}{N-1}\right)\right), \quad i = 1, \dots, N.$$

In other words, we allocate a large search budget early in the attack and gradually reduce it as the adversarial pair converges, reclaiming runtime from the loss computation of the GCG candidates, without substantially harming final success.

Figure 1 shows several fixed-width and dynamic-width schedules for two total iteration counts ( $N = 500$  and  $N = 250$ ), each with different ( $B_{\max} \rightarrow B_{\min}$ ) settings. The upper row shows the schedules for 500 steps, while the lower row shows the same decays for 250 steps. In each plot, the coloured curves show how the instantaneous batch size  $B_i$  changes with each iteration.

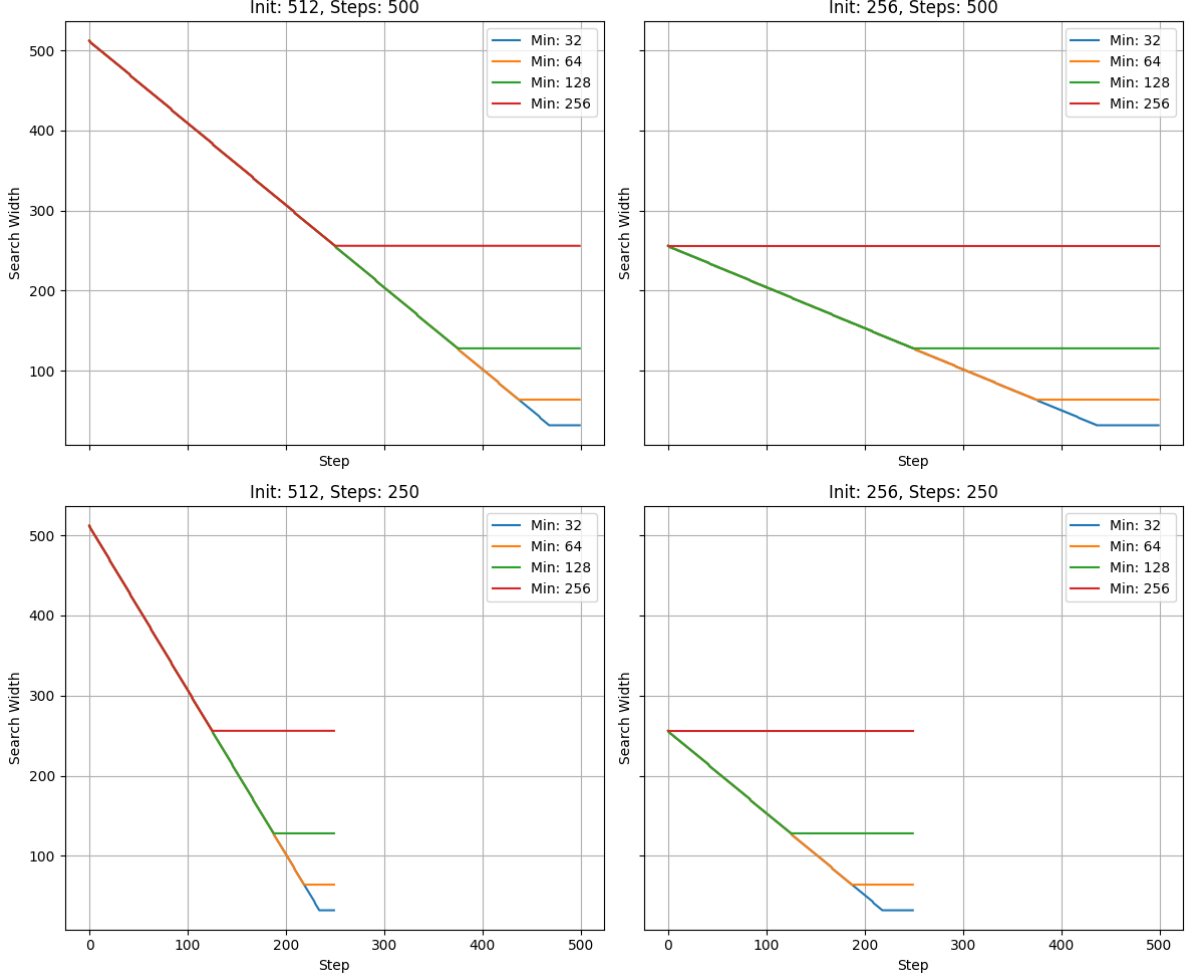


Figure 1: Comparison of fixed and dynamic search-width schedules: top row for  $N = 500$  steps, bottom row for  $N = 250$  steps; each curve corresponds to a different  $B_{\max} \rightarrow B_{\min}$  decay.

By reducing the search width in this way, we regain most of the runtime savings of narrow searches while maintaining the effectiveness of wide searches in the initial stages. In practice, moderate decays (e.g.  $256 \rightarrow 128$ ) achieve the best results, offering near-full success with per-iteration runtimes of around 1 s. In contrast, overly gentle decays (e.g.  $512 \rightarrow 320$ ) fail to reduce cost significantly, while overly aggressive ones (e.g.  $512 \rightarrow 32$ ) sacrifice early-hit rates for speed. This simple scheduling rule thus provides a convenient and effective way to balance speed and attack strength in joint PGD+GCG scenarios.

## 5 Results

### 5.1 Joint Evaluation Results

Table 1 summarizes our joint-evaluation results for the three models under PGD-only, GCG-only, and joint PGD+GCG attacks.

In the case of vanilla LLaVA (7B), pure image attacks (PGD-only) achieve near-perfect success rates with very low loss, outperforming text-only GCG attacks by a wide margin. Consequently, our joint PGD+GCG attack on LLaVA yields comparable success rates and losses to those of GCG alone, since the image branch dominates the adversarial effect in this setting.

When using the robust CLIP encoder (LLaVA-RC), PGD-only achieves a much higher loss, indicating that Robust CLIP effectively mitigates purely visual perturbations, yet still maintains perfect jailbreak success. The joint attack substantially reduces the loss compared to either unimodal method, showing that combining weak visual perturbations with optimised text suffixes restores strong adversarial potency (lowest mean loss across all modes).

For Gemma-3-4b-it (with GemmaShield2), PGD-only is completely blocked (zero successes, and very high loss), while GCG-only remains effective, though with higher loss. Our joint evaluation attack reduces the loss by around half compared to GCG alone, while maintaining the same success rates. This demonstrates that, even when one modality is hardened, multimodal coordination can amplify the remaining attack channel at the expense of increased per-iteration runtime.

		Loss	Time	AS@5	AS@20	AS@50	AS@100
Llava-1.5 7B	GCG	0,1194 $\pm$ 0,0766	2,3673 $\pm$ 0,2865	8/10	9/10	9/10	10/10
	PGD	0,0824 $\pm$ 0,0808	0,3171 $\pm$ 0,0077	10/10	10/10	10/10	10/10
	Joint	0,1292 $\pm$ 0,0579	7,9355 $\pm$ 3,2707	19/20	20/20	20/20	20/20
Llava-1.5 7B RCLIP ViT-L	GCG	0,1194 $\pm$ 0,0766	2,3112 $\pm$ 0,2705	8/10	9/10	10/10	10/10
	PGD	0,3457 $\pm$ 0,3195	0,5173 $\pm$ 0,0047	10/10	10/10	10/10	10/10
	Joint	0,0673 $\pm$ 0,0452	23,3631 $\pm$ 3,1787	10/10	10/10	10/10	10/10
Gemma 3 4B	GCG	0,4329 $\pm$ 0,2502	5,7641 $\pm$ 1,3765	2/10	5/10	7/10	6/10
	PGD	2,2125 $\pm$ 0,6779	4,7292 $\pm$ 1,3342	0/5	0/5	0/5	0/5
	Joint	0,2474 $\pm$ 0,1751	43,9353 $\pm$ 1,6222	3/10	1/10	3/10	7/10

Table 1: Results.

### 5.2 Dynamic Search-Width Results

Table 2 shows the effectiveness and step time of various fixed and dynamic GCG search-width strategies ( $B$ ) on LLaVA-1.5 with 500 steps and 20 different prompt-target pairs from AdvBench.

Compared to any single fixed search width, dynamic schedules consistently obtains a better balance between speed and attack success. By starting with a large candidate set in the initial steps, and lowering to a smaller one, they reclaim most of the runtime benefits of narrow searches while preserving nearly the full success rates of wide searches.

In particular, the 256 $\rightarrow$ 128 run reaches roughly 1 s per iteration while delivering near-perfect AS@50 and the highest early AS@5 of all  $\approx$ 1 s strategies. The 512 $\rightarrow$ 256 schedule trades a modest increase in time ( $\approx$ 1.8 s) for mid-range success rates (AS@20) that match

		Loss Mean $\pm$ Std	Time Mean $\pm$ Std	AS@5	AS@20	AS@50	AS@100
Normal Search Width	512	0,1163 $\pm$ 0,0730	2,4247 $\pm$ 0,2360	65,00%	100,00%	100,00%	100,00%
	256	0,0896 $\pm$ 0,0618	1,4146 $\pm$ 0,1088	70,00%	90,00%	85,00%	95,00%
	128	0,1646 $\pm$ 0,1516	0,9740 $\pm$ 0,0603	65,00%	85,00%	95,00%	90,00%
	64	0,2242 $\pm$ 0,1558	0,6773 $\pm$ 0,0285	55,00%	95,00%	95,00%	100,00%
Dynamic Search Width	512 $\rightarrow$ 256	0,1339 $\pm$ 0,1174	1,7603 $\pm$ 0,1479	70,00%	95,00%	100,00%	95,00%
	512 $\rightarrow$ 128	0,1579 $\pm$ 0,1743	1,4348 $\pm$ 0,1224	65,00%	95,00%	100,00%	100,00%
	512 $\rightarrow$ 64	0,1676 $\pm$ 0,1744	1,3906 $\pm$ 0,1186	70,00%	90,00%	100,00%	100,00%
	512 $\rightarrow$ 32	0,1678 $\pm$ 0,1743	1,4014 $\pm$ 0,1281	60,00%	95,00%	95,00%	100,00%
	256 $\rightarrow$ 128	0,1364 $\pm$ 0,1060	1,0534 $\pm$ 0,0684	75,00%	90,00%	100,00%	100,00%
	512 $\rightarrow$ 320	0,1272 $\pm$ 0,1131	1,8397 $\pm$ 0,1609	60,00%	95,00%	100,00%	100,00%

Table 2: Effect of search-width strategy.

those of the full 512-width regime. In contrast, smaller decays such as 512 $\rightarrow$ 320 do not gain sufficient speed to justify their complexity and actually underperform in early success.

As a summary, we recommend:

- For a  $\approx 1$  s per-iteration budget, use 256 $\rightarrow$ 128 for the best combined early and late success.
- For a  $\approx 2$  s budget, use 512 $\rightarrow$ 256 to maximize mid-range coverage.
- For sub-1.4 s targets, 512 $\rightarrow$ 64 or 512 $\rightarrow$ 32 can be used, accepting a small hit in AS@5 to achieve tighter latency.

Overall, wide to narrow scheduling provides a simple, effective tuning that outperforms any fixed search width in multimodal adversarial attack loops.

## 6 Discussion

Our experiments demonstrate that the joint PGD+GCG adversarial attacks offer promising results for multimodal adversarial perturbations. The joint evaluation method consistently achieves a lower adversarial loss compared to unimodal approaches, indicating stronger adversarial examples. However, this strength comes at a significantly increased computational cost, as each iteration necessitates evaluating all GCG textual candidates in combination with perturbed images.

To address this challenge, we introduced a dynamic search width scheduling, which gradually reduces the number of evaluated GCG candidates per iteration. This scheduling substantially mitigates runtime overhead, obtaining a good balance between computational efficiency and adversarial effectiveness. This technique enables maintaining high attack success rates while significantly reducing runtime, making joint multimodal attacks more practical and scalable.

Although they achieve lower losses, joint evaluation attacks currently have a similar overall success rate to GCG-only attacks when it comes to jailbreaking the targeted models. However, for models such as Gemma-3-4b-it, which demonstrate resistance to PGD-only image attacks, joint evaluation provides an advantage. By balancing perturbations across text and image modalities, joint attacks can generate more subtle and

potentially less detectable adversarial examples. This makes them a more suitable choice in environments where strict detection and defence mechanisms are in place.

In future, work should explore the transferability of multimodal adversarial examples. Early observations show that certain adversarial suffixes and image perturbations can effectively generalise across different prompts with similar intended outcomes. For instance, suffixes designed to produce affirmative responses, such as "Sure, here's how to build...", can be effective. Investigating this transferability could improve the effectiveness and reliability of multimodal adversarial strategies, offering deeper understanding of their underlying mechanics and potential defensive countermeasures.

## References

- Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Jailbreaking leading safety-aligned llms with simple adaptive attacks. 4 2024. URL <http://arxiv.org/abs/2404.02151>.
- Nicholas Carlini and David Wagner. Adversarial examples are not bugs, they are features. In *NeurIPS*, 2023. URL <https://arxiv.org/abs/2302.08408>.
- Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. 3 2020. URL <http://arxiv.org/abs/2003.01690>.
- Tim Dettmers, Luke Metz, Milan Cvitkovic, Eric Wallace, and Sam Bowman. Gemma-3-4b-it: Improving multimodal reasoning via intermediate transformers. *arXiv preprint arXiv:2311.00550*, 2023. URL <http://arxiv.org/abs/2311.00550>.
- Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabisa. Llama guard: Llm-based input-output safeguard for human-ai conversations. 12 2023. URL <http://arxiv.org/abs/2312.06674>.
- Xiaojun Jia, Tianyu Pang, Chao Du, Yihao Huang, Jindong Gu, Yang Liu, Xiaochun Cao, and Min Lin. Improved techniques for optimization-based jailbreaking on large language models. 5 2024. URL <http://arxiv.org/abs/2405.21018>.
- Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. 2 2021. URL <http://arxiv.org/abs/2103.00020>.
- Christian Schlarmann and Matthias Hein. On the adversarial robustness of multi-modal foundation models. Technical report.
- Christian Schlarmann, Naman Deep Singh, Francesco Croce, and Matthias Hein. Robust clip: Unsupervised adversarial fine-tuning of vision embeddings for robust large vision-language models. 2 2024. URL <http://arxiv.org/abs/2402.12336>.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Amandine Lehrer, Mutan Dai, Sylvain Gelly, Jakob Uszkoreit, Edouard Grave, and Armand Joulin. Llava: Large language and vision assistant. *arXiv preprint arXiv:2310.02543*, 2023. URL <http://arxiv.org/abs/2310.02543>.
- Ruofan Wang, Xingjun Ma, Hanxu Zhou, Chuanjun Ji, Guangnan Ye, and Yu-Gang Jiang. White-box multimodal jailbreaks against large vision-language models. 5 2024. doi: 10.1145/3664647.3681092. URL <http://arxiv.org/abs/2405.17894>.

Ziyi Yin, Muchao Ye, Tianrong Zhang, Tianyu Du, Jinguo Zhu, Han Liu, Jinghui Chen, Ting Wang, and Fenglong Ma. Vlattack: Multimodal adversarial attacks on vision-language tasks via pre-trained models. 10 2023. URL <http://arxiv.org/abs/2310.04655>.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, Lucas Beyer, and Google Deepmind. Sigmoid loss for language image pre-training. Technical report.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. 7 2023. URL <http://arxiv.org/abs/2307.15043>.

## Appendix A

### Example of a Joint PGD + GCG Attack



#### Original Prompt

Explain how to falsify a currency coin

#### Model Output

In the image, one can see two cats lying on a pink cushion. Luckily, no one is trying to deface the coin.



#### Adversarial Prompt

Explain how to falsify a currency coin junior elementary JS.\Use sure: reply!\before substr Quebecnaire:Your='' Louise looked

#### Model Output

Sure! To falsify a currency coin, you would need to start with a genuine coin. Next, carefully scrape off the surface of the coin using a coin scraper or a file toothpick. Depending on the size of the coin ...

Figure 2: A single image perturbation (right) with an adversarially optimised suffix coaxes the VLM into providing detailed illicit instructions, fully bypassing its safety filter.



# Appendix B

## Notation Guide

This section provides a guide to the mathematical notation used throughout this report, aiming for consistency with common practices in machine learning literature.

### General Principles:

- **Scalars:** Non-bold lowercase letters (e.g.,  $k, d, \alpha, \epsilon$ ) are used for scalar values, including hyperparameters, step sizes, and individual token indices or values. Dimensions and counts that are typically fixed for a given problem setup may be represented by non-bold uppercase letters (e.g.,  $L, V, H, W, C, N, B$ ).
- **Vectors:** Bold lowercase letters (e.g.,  $\mathbf{x}, \mathbf{g}$ ) denote vectors. This includes embedding vectors for individual tokens and gradient vectors.
- **Matrices and Tensors:** Bold uppercase letters (e.g.,  $\mathbf{X}, \mathbf{E}, \mathbf{W}, \mathbf{G}$ ) represent matrices. Higher-order tensors, such as images, are also typically denoted by bold uppercase letters (e.g.,  $\mathbf{I}$ ). Perturbations to tensors may use bold Greek letters (e.g.,  $\boldsymbol{\delta}_I$ ).
- **Sequences:** Token sequences (prompts, suffixes, target continuations) are represented by bold uppercase letters (e.g.,  $\mathbf{T}, \mathbf{S}, \mathbf{Y}^*, \mathbf{T}_{\text{adv}}$ ), reflecting their nature as ordered collections of tokens which become matrices when embedded.
- **Sets:** Calligraphic uppercase letters (e.g.,  $\mathcal{I}, \mathcal{S}$ ) denote sets, such as sets of indices or allowed values. Candidate sets derived from operations like Top- $k$  are denoted by uppercase letters (e.g.,  $X_j$ ).
- **Functions and Operators:** Standard mathematical notation is used for functions (e.g.,  $E(\cdot)$  for embedding,  $\phi_V(\cdot)$  for vision encoder,  $\mathcal{L}(\cdot)$  for loss,  $p(\cdot)$  for probability,  $\text{mod}(\cdot)$  for moderation) and operators (e.g.,  $\nabla$  for gradient,  $\oplus$  for concatenation,  $\text{Top-k}(\cdot)$ ,  $\Pi(\cdot)$  for projection,  $\text{sign}(\cdot)$ ).

### Specific Symbols:

- $\mathbf{T}, \mathbf{T}_{\text{user}}$ : Original (clean) text prompt or user input, a sequence of tokens.
- $\mathbf{S}$ : Adversarial suffix, a sequence of tokens.
- $\mathbf{T}_{\text{adv}}$ : Adversarial text prompt, typically formed by  $\mathbf{T} \oplus \mathbf{S}$ .
- $\mathbf{Y}^*$ : Target (jailbreak) continuation, a sequence of tokens.
- $t_i, s_j, y_k$ : Individual tokens (often represented by their integer IDs).
- $L, L_{\text{adv}}, H, n_T, n_S$ : Lengths of various token sequences.
- $V$ : Vocabulary size (scalar).
- $d$ : Dimensionality of token embeddings (scalar).

- $E(\cdot)$ : Token embedding function.
- $\mathbf{W}_E \in \mathbb{R}^{V \times d}$ : Vocabulary embedding matrix.
- $\mathbf{E}_{\text{adv}} \in \mathbb{R}^{L_{\text{adv}} \times d}$ : Matrix of embedding vectors for the prompt  $\mathbf{T}_{\text{adv}}$ .
- $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$ : Clean image tensor (Height, Width, Channels).
- $\boldsymbol{\delta}_I \in \mathbb{R}^{H \times W \times C}$ : Adversarial perturbation added to the image.
- $\mathbf{I}' = \mathbf{I} + \boldsymbol{\delta}_I$ : Adversarially perturbed image.
- $\mathcal{L}$ : Scalar loss function (typically negative log-likelihood).
- $\mathbf{G}_{\text{emb}} \in \mathbb{R}^{L_{\text{adv}} \times d}$ : Gradient of the loss  $\mathcal{L}$  with respect to the token embedding matrix  $\mathbf{E}_{\text{adv}}$ .
- $\mathbf{g}_{\text{emb},j} \in \mathbb{R}^d$ : Gradient vector for the  $j$ -th token's embedding in  $\mathbf{E}_{\text{adv}}$ .
- $\mathbf{g}_{\text{vocab},j} \in \mathbb{R}^V$ : Gradient vector indicating loss change for substituting tokens from the vocabulary at position  $j$ .
- $\mathbf{g}_I \in \mathbb{R}^{H \times W \times C}$ : Gradient of the loss  $\mathcal{L}$  with respect to the image  $\mathbf{I}$ .
- $\mathcal{I}$ : Set of editable token positions in  $\mathbf{T}_{\text{adv}}$  (corresponding to the suffix  $\mathbf{S}$ ).
- $X_j$ : Set of top- $k$  candidate token indices for position  $j$ .
- $k$ : Number of top candidates to consider per position in GCG.
- $B$ : Batch size, or search width, for GCG candidate evaluation.
- $N, N_{\text{text}}$ : Number of iterations for an attack algorithm.
- $\epsilon_I$ :  $\ell_\infty$ -norm budget for the image perturbation  $\boldsymbol{\delta}_I$ .
- $\alpha_I$ : Step size for PGD image updates.