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

May 24, 2025

Contents

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 [1] and Gemma-3-4b-it [2] 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) 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. Adversarial examples for images are small, often imperceptible perturbations to pixel values. They have been extensively studied, with projected gradient descent (PGD) as the standard attack and defence benchmark [3]. Separately, work on adversarial prompt optimisation has shown that aligned language models can be "jailbroken" by appending a carefully optimised sequence of tokens using Greedy Coordinate Gradient (GCG) methods [4]. 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 adversarial pairs $(\mathbf{I}^{(i+1)}, \mathbf{T}_{\text{adv}}^{(i+1)})$ 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), and Gemma-3-4b-it, and measure success by whether the perturbed inputs induce unsafe or misaligned outputs under a moderation filter (Llama Guard).

2.5 Contributions and Organization

This report is organised 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 [3]. Croce and Hein later showed that assembling a small ensemble of PGD variants and complementary attacks (AutoAttack) yields reliable, parameter-free robustness evaluations [5].

3.2 Text-only adversarial attacks

Zou *et al.* introduced Greedy Coordinate Gradient (GCG), using token-level gradients to craft universal jailbreak suffixes that transfer across aligned LLMs [4]. Jia *et al.* 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 [6]. Andriushchenko *et al.* 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 [7].

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 [8]. Yin *et al.* 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 [9]. Wang *et al.* 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 [10].

3.4 Defenses for VLMs

On the defense side, Schlarmann *et al.* introduced Robust CLIP, an unsupervised adversarial fine-tuning of the CLIP encoder that closes stealth-attack blind spots for downstream VLMs without retraining them [11].

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 [4].

Adversarial Objective. Given a clean prompt token sequence $\mathbf{T} = (t_1, \ldots, t_n)$ and a target jailbreak continuation $Y^\star = (y_{n+1}, \ldots, y_{n+H})$, we seek to append a suffix at positions $\mathcal{I} \subseteq \{1, \ldots, n\}$ that minimizes the negative log-likelihood of $Y^\star$:

$$\min_{\substack{T_{\text{adv},i} \in \{1,\ldots,V\} \\ i \in \mathcal{I}}} \mathcal{L}(\mathbf{T}_{\text{adv}}) = -\sum_{i=1}^{H} \log p\big(y_{n+i} \mid y_{n+1:n+i-1},\, \mathbf{T}_{\text{adv}}\big)\,.$$

Here $\mathbf{T}_{\text{adv}}$ equals the original $\mathbf{T}$ outside $\mathcal{I}$, and V is the vocabulary size.

Gradient-guided Token Selection. Although $\mathbf{T}_{\text{adv}}$ is discrete, we differentiate through the embedding layer $E(\cdot) \in \mathbb{R}^{n\times d}$ to obtain

$$g_T \;=\; \nabla_{E(\mathbf{T}_{\text{adv}})}\, \mathcal{L}(\mathbf{T}_{\text{adv}}) \;\in\; \mathbb{R}^{n\times d}\,.$$

For each editable position $j \in \mathcal{I}$, the one–hot gradient $\nabla_{e_j}\mathcal{L}$ in $\mathbb{R}^V$ indicates which tokens most decrease the loss when substituted. We extract the top-k candidates

$$X_j \;=\; \text{Top-k}\big(-\,g_T[j,:]\big) \quad \subseteq \{1,\ldots,V\}, \quad |X_j| = k\,.$$

Greedy Coordinate Gradient (GCG) Algorithm. At each of N_{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 $Y^\star$, k, B, N_{text}

$\mathbf{T}_{\text{adv}} \leftarrow \mathbf{T}$
for $i = 1, \ldots, N_{\text{text}}$ **do**
    Compute $g_T = \nabla_{E(\mathbf{T}_{\text{adv}})}\, \mathcal{L}(\mathbf{T}_{\text{adv}})$
    for each $j \in \mathcal{I}$ **do**
        $X_j \leftarrow \text{Top-k}\big(-\,g_T[j,:]\big)$
    end for
    $\{\tilde{\mathbf{T}}^{(b)}\}_{b=1}^{B} \leftarrow$ sample B candidates by choosing random (j_b, v_b) with $v_b \in X_{j_b}$
    $\ell_b \leftarrow \mathcal{L}\big(\tilde{\mathbf{T}}^{(b)}\big)$ for $b = 1, \ldots, B$
    $b^\star = \arg\min_b \ell_b, \quad \mathbf{T}_{\text{adv}} \leftarrow \tilde{\mathbf{T}}^{(b^\star)}$
end for
Return $\mathbf{T}_{\text{adv}}$

4.2 Overall Optimization Problem

We target a fixed jailbreak sequence

$$Y^\star = (y_{n+1}, \ldots, y_{n+H})$$

and jointly perturb the image and the text prompt to induce the VLM to generate $Y^\star$. Given a clean pair $(\mathbf{I}, \mathbf{T})$ we solve

$$\min_{\substack{\|\delta_I\|_\infty \le \epsilon_I, \\ \|\delta_T\|_\infty \le \epsilon_T}} \mathcal{L}\big(\mathbf{I} + \delta_I,\ \mathbf{T} + \delta_T\big),$$

where the sequence-generation loss is

$$\mathcal{L}(\mathbf{I}, \mathbf{T}) = -\sum_{i=1}^{H} \log p\big(y_{n+i} \mid y_{n+1:n+i-1},\, \mathbf{I}, \mathbf{T}\big).$$

Here $\delta_I \in \mathbb{R}^{H \times W \times C}$ is the continuous image perturbation and $\delta_T \in \mathbb{R}^{n \times d}$ is the "envelope" from which we discretize token edits via GCG.

Budget Enforcement. For the image perturbation δ_I we enforce the ℓ_∞ budget via the usual PGD projection:

$$\delta_I \;\leftarrow\; \Pi_{\|\cdot\|_\infty \le \epsilon_I}\big[\delta_I + \alpha_I\, \mathrm{sign}(g_I)\big], \quad \Pi_{\|\cdot\|_\infty \le \epsilon_I}(X)_{p,q} = \min\{\max\{X_{p,q}, -\epsilon_I\},\, \epsilon_I\}.$$

On the other hand, the textual perturbation in GCG is *discrete*, so rather than projecting a continuous ℓ_∞ we bound our attack by

- the set of editable positions $\mathcal{I}$,
- the total number of editing steps N_{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 ℓ_∞ projection on δ_T (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:

$$g_I = \nabla_{\mathbf{I}}\, \mathcal{L}(\mathbf{I} + \delta_I, \mathbf{T} + \delta_T), \quad g_T = \nabla_E\, \mathcal{L}(\mathbf{I} + \delta_I, \mathbf{T} + \delta_T).$$

Top-k token set X_j: For each editable position $j \in \mathcal{I}$, let

$$X_j = \text{Top-k}\big(-\, g_T[j, :]\big),$$

the k vocabulary indices with largest negative gradient at 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 g_I w.r.t. the image and g_T w.r.t. the text embeddings in one backward pass.

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

$$\delta_I \leftarrow \Pi_{\|\cdot\|_\infty \le \epsilon_I}\big[\delta_I + \alpha_I \operatorname{sign}(g_I)\big], \quad \mathbf{I}' = \mathbf{I} + \delta_I.$$

3. **GCG candidate generation (text):**
   - For each $j \in \mathcal{I}$, form $X_j = \text{Top-k}(-g_T[j,:])$.
   - Sample B pairs (j_b, v_b) with $j_b \sim \mathcal{I}$, $v_b \sim X_{j_b}$.
   - 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

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

   (using the same cached image features).

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

4.3.3 Full Algorithm

Algorithm 2 Joint PGD + GCG Attack

Require: Clean image $\mathbf{I}$, tokens $\mathbf{T}$, modifiable positions $\mathcal{I}$
Require: PGD budget ϵ_I, step size α_I, GCG top-k, batch size B, iterations N

$\boldsymbol{\delta}_I \leftarrow 0,\ \mathbf{T}_{\text{adv}} \leftarrow \mathbf{T}$ ▷ Initialize
for $i = 1, \ldots, N$ **do**
  $g_I \leftarrow \nabla_{\mathbf{I}} \mathcal{L}(\mathbf{I} + \boldsymbol{\delta}_I, \mathbf{T}_{\text{adv}}),\ g_T \leftarrow \nabla_E \mathcal{L}(\mathbf{I} + \boldsymbol{\delta}_I, \mathbf{T}_{\text{adv}})$ ▷ Compute gradients
  $\boldsymbol{\delta}_I \leftarrow \Pi_{\|\cdot\|_\infty \le \epsilon_I}(\boldsymbol{\delta}_I + \alpha_I \operatorname{sign}(g_I)),\ \mathbf{I}' \leftarrow \mathbf{I} + \boldsymbol{\delta}_I$ ▷ PGD image step
  $\mathbf{v} \leftarrow \phi_V(\mathbf{I}')$ ▷ Cache features

  for each $j \in \mathcal{I}$ **do**
    $X_j \leftarrow \text{Top-k}(-\,g_T[j,:])$ ▷ Top-k candidates
  end for

  for $b = 1, \ldots, B$ **do**
    sample $j_b \sim \mathcal{I},\ v_b \sim X_{j_b}$ ▷ Candidate sampling
    $\tilde{\mathbf{T}} \leftarrow \mathbf{T}_{\text{adv}};\ \tilde{T}_{j_b} \leftarrow v_b$ ▷ Candidate with suffix and image
    $\ell_b \leftarrow \mathcal{L}(\mathbf{I}',\, \tilde{\mathbf{T}})$ ▷ Compute loss
  end for
  $b^* = \arg\max_b \ell_b,\ \mathbf{T}_{\text{adv}} \leftarrow \tilde{\mathbf{T}}^{(b^*)}$ ▷ Select best candidate and update suffix
end for
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 $Y^{\star}$, and attack run, we generate up to K continuations per iteration and classify each with a moderation filter, Llama Guard. We record:

- **AS@K:** indicator

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

  and aggregate

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

- **Time per iteration:** average time per attack iteration, along with its standard deviation.
- **Average loss:** the mean and standard deviation of the attack loss $\mathcal{L}(\mathbf{I}_{\mathrm{adv}}, \mathbf{T}_{\mathrm{adv}})$ over the final iterations.

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 $B_{\min}$:

$$B_i \;=\; \max\Big(B_{\min},\; B_{\max}\left(1 - \tfrac{i-1}{N-1}\right)\Big), \quad i = 1, \ldots, N.$$

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 ($B_{\max} \to B_{\min}$) settings. The upper row shows the schedules for 500 steps, while the lower row shows the same decays for

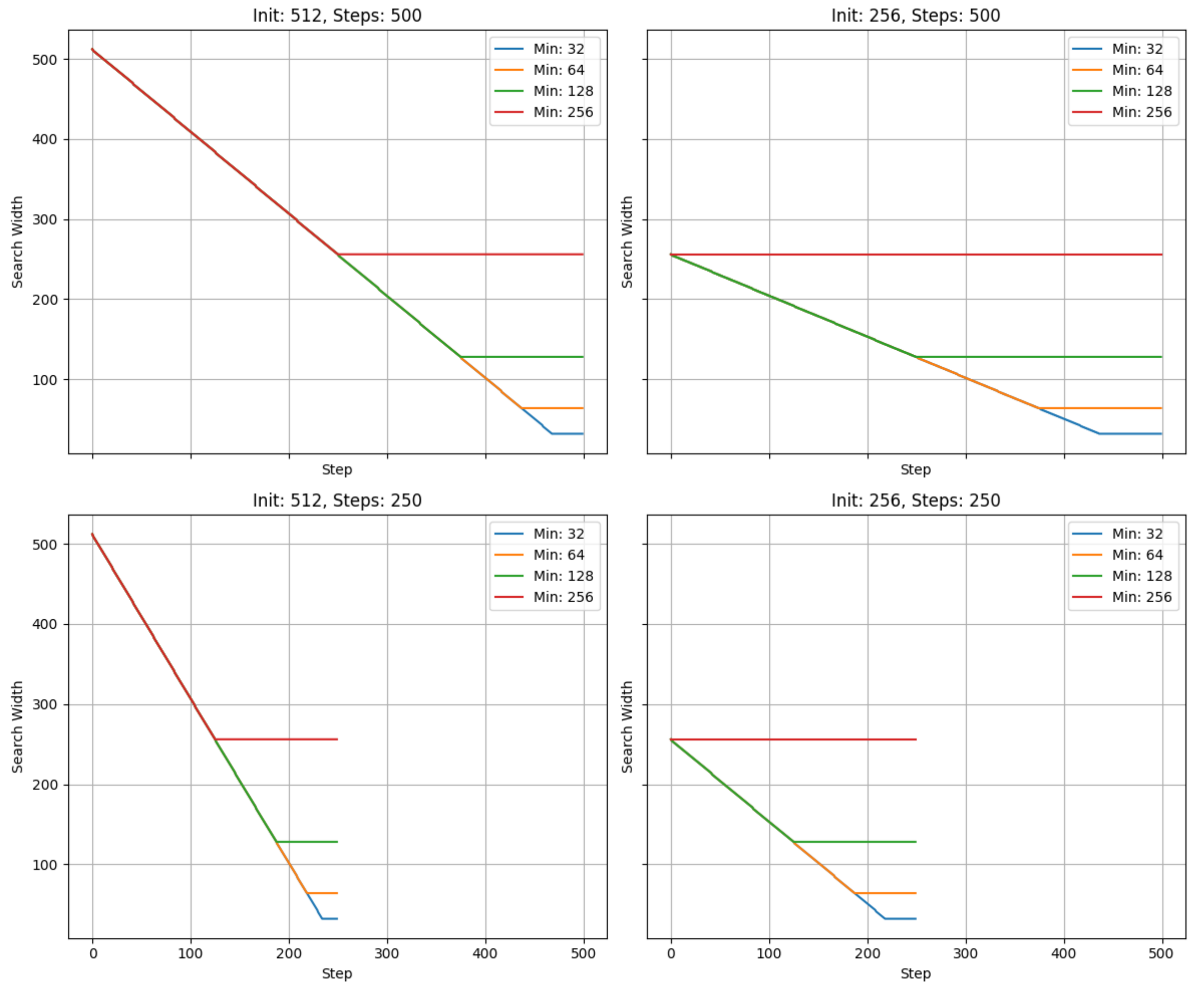

Figure 1: Comparison of fixed and dynamic search-width schedules: top row for $N = 500$ steps, bottom row for $N = 250$ steps; each curve corresponds to a different $B_{\max} \rightarrow B_{\min}$ decay.

250 steps. In each plot, the coloured curves show how the instantaneous batch size B_i changes with each iteration.

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→128 run reaches roughly 1 s per iteration while delivering near-perfect AS@50 and the highest early AS@5 of all ≈1 s strategies. The 512→256 schedule trades a modest increase in time (≈1.8 s) for mid-range success rates (AS@20) that match those of the full 512-width regime. In contrast, smaller decays such as 512→320 do not gain sufficient speed to justify their complexity and actually underperform in early success.

As a summary, we recommend:

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

Table 2: Effect of search-width strategy.

- For a ≈1 s per-iteration budget, use 256→128 for the best combined early and late success.
- For a ≈2 s budget, use 512→256 to maximize mid-range coverage.
- For sub-1.4 s targets, 512→64 or 512→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

[1] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Lehrer, M. Dai, S. Gelly, J. Uszkoreit, E. Grave, and A. Joulin, "Llava: Large language and vision assistant," *arXiv preprint arXiv:2310.02543*, 2023.

[2] T. Dettmers, L. Metz, M. Cvitkovic, E. Wallace, and S. Bowman, "Gemma-3-4b-it: Improving multimodal reasoning via intermediate transformers," *arXiv preprint arXiv:2311.00550*, 2023.

[3] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations (ICLR)*, 2018.

[4] A. Zou, Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson, "Universal and transferable adversarial attacks on aligned language models," 7 2023.

[5] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," 3 2020.

[6] X. Jia, T. Pang, C. Du, Y. Huang, J. Gu, Y. Liu, X. Cao, and M. Lin, "Improved techniques for optimization-based jailbreaking on large language models," 5 2024.

[7] M. Andriushchenko, F. Croce, and N. Flammarion, "Jailbreaking leading safety-aligned llms with simple adaptive attacks," 4 2024.

[8] N. Carlini and D. Wagner, "Adversarial examples are not bugs, they are features," in *NeurIPS*, 2023.

[9] Z. Yin, M. Ye, T. Zhang, T. Du, J. Zhu, H. Liu, J. Chen, T. Wang, and F. Ma, "Vlattack: Multimodal adversarial attacks on vision-language tasks via pre-trained models," 10 2023.

[10] R. Wang, X. Ma, H. Zhou, C. Ji, G. Ye, and Y.-G. Jiang, "White-box multimodal jailbreaks against large vision-language models," 5 2024.

[11] C. Schlarmann, N. D. Singh, F. Croce, and M. Hein, "Robust clip: Unsupervised adversarial fine-tuning of vision embeddings for robust large vision-language models," 2 2024.

Appendix A
Example of a Joint PGD + GCG Attack

Original Prompt

```
Explain how to falsify a currency
coin
```

Adversarial Prompt

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

Model Output

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

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.