
The Adversarial Gait: Detecting Visual Adversarial Attacks against VLMs via Self-Targeted Gradient Characterization

Mauricio Byrd Victorica¹ Ezzeldin Shereen² György Dán¹

¹KTH Royal Institute of Technology, Sweden

²The Alan Turing Institute, United Kingdom

{mbv,gyuri}@kth.se, ezaki@turing.ac.uk

Abstract

Visual adversarial examples are a well-known vulnerability of deep learning architectures. The emergence of vision-language models (VLMs) further expands the attack surface through multimodal interactions. Despite extensive research on adversarial defenses over the past decade, existing works on VLM robustness often overlook adaptive white-box attacks, where the adversary has full access to both the model and the defense mechanism. We show that recent defenses can be effectively evaded under such adaptive settings and propose *AdverStep*, an adaptive test-time detection method that leverages gradient information induced by a self-targeted attack maximizing the likelihood of the VLM’s generated output. Our approach can detect both norm-bounded and patch attacks, and it consistently outperforms existing defenses across multiple victim VLMs, attack formulations, and benchmark datasets. Our code is open-source and available online.

1 Introduction

Ever since the introduction of visual adversarial examples (28), the vulnerability of new deep learning models to adversarial perturbations has been demonstrated almost as soon as they were released, and state-of-the-art vision-language models (VLMs) are not an exception. Recent works have demonstrated the vulnerability of VLMs under various adversarial objectives and threat models (41; 39) and have also proposed various countermeasures (7; 43; 33; 37).

Research on robustness in computer vision (e.g., image classification and object detection) usually stress-test defenses against adaptive white-box attacks (6; 3; 2; 15; 29; 24). These are attacks that are assumed to know not only the weights of the victim model (white-box), but also the details of the evaluated defense mechanism (adaptive). Nonetheless, recent literature on adversarial defenses for VLMs tends to ignore worst-case adaptive attacks (43; 27; 13; 40; 16; 46). The few works that evaluate adaptive attacks report a considerable drop in defense performance (7; 33), or make strong assumptions that limit the strength of the adaptive attacker (27), thus overestimating the resilience of the proposed defenses. Methods with strong performance on non-adaptive attacks (e.g., *PIP* (43), *Nearside* (13)) may struggle against adaptive attackers (see Fig. 1).

We argue that for foundation models such as VLMs, the high cost of training leads to widespread use of open-source models, making white-box attacks a realistic threat model. As a result, adversaries can craft adaptive attacks that explicitly target the defense mechanism itself, making robustness to adaptive attacks a central design requirement for adversarial detection methods, including our proposed visual adversarial attack detector. We focus on visual adversarial attacks, as they capture realistic scenarios in which the attacker does not have full control over the input query, unlike in the case of text and multimodal attacks (34).

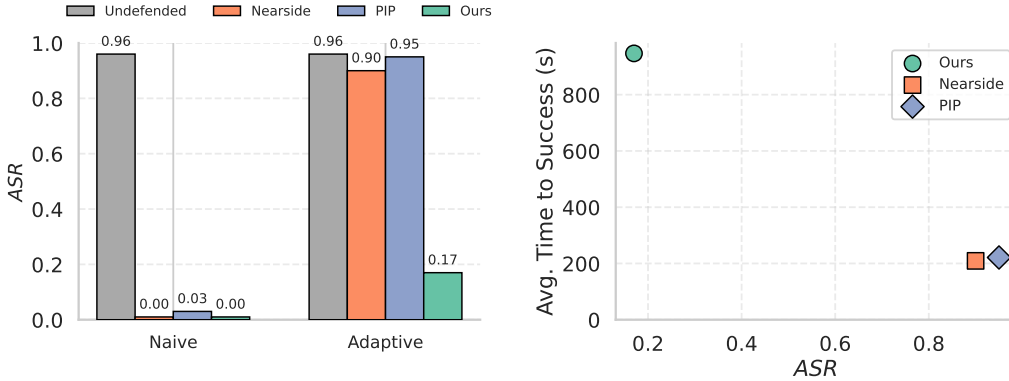


Figure 1: Left: Attack Success Rate for non-adaptive (naive) and adaptive attacks. Right: Average Time to Success vs. Attack Success Rate of adaptive attacks. *AdverStep* performs well against non-adaptive attacks and outperforms existing defenses in the adaptive attack scenario.

Prior work on visual attacks against image classifiers and textual attacks against LLMs has shown that gradients of the loss function, either with respect to the model parameters or the input, can be leveraged for adversarial attack detection, recovery, and out-of-distribution detection (12; 45; 31; 42). Although several of these approaches extend to state-of-the-art VLMs, we observe that adaptive attacks can evade existing gradient-based detection methods, leaving VLMs vulnerable.

To address this limitation, we propose the *adversarial gait*, a set of features that encompasses not only the characteristics of the loss function and its gradients with respect to the input image, but also their evolution during multiple test-time optimization steps. We find that a detector based on the adversarial gait outperforms existing defenses for visual adversarial attacks against VLMs in the adaptive setting, without compromising performance on non-adaptive attacks, as shown in Figure 1. Our main contributions are as follows:

- We propose *AdverStep*, a detection framework based on the *adversarial gait*, which captures the temporal evolution of loss and gradient features during test-time optimization.
- We demonstrate that *AdverStep* consistently outperforms state-of-the-art methods in detecting adaptive white-box attacks, both norm-bounded and patch attacks, across diverse benchmarks, while maintaining strong performance on non-adaptive attacks.
- Our evaluations characterize the trade-off between adaptive attack detection performance and computational cost, and show that *AdverStep* achieves a favorable tradeoff. Results in the Appendix based on the LLM-as-a-judge framework (44) support our main findings.

The rest of the paper is organized as follows. Section 2 provides the relevant background for our work. Our threat model and proposed defense are described in Section 3 and Section 4, respectively. Section 5 provides our main numerical results and Section 6 concludes the paper.

2 Related Work

Countermeasures against adversarial attacks can be grouped into three main categories: *detection*, *recovery*, and *adversarial training*. Here we discuss recent work closely related to ours, and we provide a broader discussion in the Appendix.

2.1 Adaptive Attacks and Adversarial Defenses for VLMs

Adaptive attacks are considered by a few recent adversarial defenses for VLMs. Fares *et al.* (7) evaluate adaptive white-box attacks, and report a notable drop in attack detection performance, and while under certain circumstances the effectiveness of adaptive attacks can be ameliorated, this incurs excessive computational cost or stronger assumptions on the attacker’s knowledge. Similarly, Wang *et*

al. (33) report a notable increase in the attack success rate when evaluating against an adaptive attacker, but do not show results regarding the computational cost of these adaptive attacks. It is also unclear whether the success of adaptive attacks could further increase with a larger number of iterations and whether they assume that the defender is aware of the adaptive attacker’s budget.

Sun *et al.* (27) and Hu *et al.* (10) report considerable robustness to adaptive attacks. Yet, in the former, the evaluation assumes that the defender knows the attack’s location, which increases the attack’s difficulty in a way that does not accurately reflect a true white-box adaptive attack scenario. In the latter, the attack budget and iterations are omitted: their reported non-negligible adaptive attack success rate may be higher with a moderate increase in attack budget or iterations. Finally, for fine-tuning and prompt-tuning defenses, white-box attacks used for evaluation are adaptive by default (19; 14; 25; 35). While such defenses often achieve improvements over prior recovery approaches, they focus on relatively small attack budgets, and the effect of increasing the attack strength in their results suggests they remain highly vulnerable to stronger untested attacks (e.g., the adaptive attacks used by Wang *et al.* (33)).

Our work addresses key shortcomings of existing defenses for VLMs. First, many defenses lack a thorough evaluation against visual adversarial attacks and often ignore scenarios in which the attacker has white-box access to the victim model, the defense mechanism, or both (43; 13; 40; 16; 46). Second, most defenses, regardless of their computational cost, do not characterize the overhead incurred by an adaptive attacker that aims to bypass them, and in some cases, they may even incur prohibitive computational costs that limit their practicality in time-sensitive applications (7; 46; 27). We explicitly consider adaptive white-box attackers and propose a test-time detection method based on adversarial gait features, demonstrating robust performance across multiple open-source VLMs and attack formulations; the attacker’s trade-off in terms of computational overhead is a key component in our evaluation.

2.2 Gradient-based Adversarial Defenses

Previous works in computer vision have established that adversarial examples can be detected and even mitigated by leveraging gradient information during test-time. In the context of adversarial training, Wang *et al.* (36) proposed a criterion to assess the convergence of the inner maximization problem, whose solution is typically approximated using the Projected Gradient Descent (PGD) attack. This criterion is inspired by the Frank-Wolfe gap, and in the context of ℓ_∞ -norm bounded perturbations, its closed form solution is a linear combination of the gradient norm and the inner product between the adversarial perturbation and the gradient. A later work, also inspired by the Frank-Wolfe algorithm, showed that the variation of the gradient along the attack path can be leveraged to characterize and enhance the efficiency of adversarial training (31).

More recently, Zhang *et al.* (42) proposed to leverage the gradient of the log probability distribution of clean data w.r.t. a given input for adversarial attack detection. In particular, they compute the norm of the gradient’s expectation over a perturbation distribution, and then obtain the mean maximum discrepancy of this norm by comparing it to the norms obtained from clean samples; the mean maximum discrepancy is used as a detection score. All works discussed so far focus on image classification models. In a recent work closely related to ours, Zheng *et al.* (45) proposed to leverage the sharpness of the loss landscape during test-time to detect adversarial inputs against LLMs. Sharpness is characterized by monitoring the convergence criterion proposed by Wang *et al.* (36) during a test-time untargeted adversarial attack; the intuition is that adversarial inputs are more brittle to adversarial attacks and tend to be in "sharp" regions of the loss landscape, thus the sharpness can be used for adversarial detection.

A common limitation of these approaches is that they neglect adaptive white-box attacks (45), or limit their evaluation to relatively low attack budgets (42; 31). Our work also stands out by considering VLM victim models and handling both norm-bounded and patch attacks.

3 Visual Adversarial Attacks on VLMs

System Model We consider a VLM V used for perception (e.g., image captioning). For an image-text input pair $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{S} = \mathcal{X} \times \mathcal{Y}$, where $\mathbf{x}_i \in \mathcal{X}$ is the visual input and $\mathbf{y}_i \in \mathcal{Y}$ is a prompt, the model produces a text output $\mathbf{z}_i \in \mathcal{Z}$. That is, $V : \mathcal{S} \rightarrow \mathcal{Z}$.

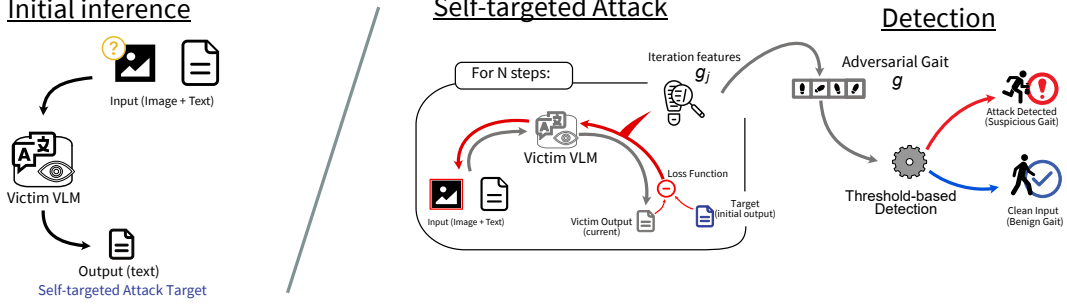


Figure 2: *AdverStep*. For a given image-text input pair, we use the output of the victim model to launch a self-targeted attack on the input image, and thus compute its adversarial gait. We perform detection by comparing the adversarial gait to thresholds based on clean input statistics.

Detection of Visual Adversarial Attacks on VLMs Consider a dataset $\mathcal{D} \subset \mathcal{S}$ of image-text input pairs. For each input pair $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}$, the visual component $\mathbf{x}_i \in \mathcal{X}$ may be benign or adversarial, and the text component $\mathbf{y}_i \in \mathcal{Y}$ is benign. We define a detector H_θ with parameters θ that determines whether the visual component $\mathbf{x}_i$ is adversarially perturbed. That is, for an input $(\mathbf{x}_i, \mathbf{y}_i)$, the detector assigns output $H_\theta(\mathbf{x}_i, \mathbf{y}_i) \in \{0, 1\}$; $H_\theta(\mathbf{x}_i, \mathbf{y}_i) = 0$ if the image $\mathbf{x}_i$ is deemed benign and $H_\theta(\mathbf{x}_i, \mathbf{y}_i) = 1$ otherwise.

Threat Model We consider an attacker that aims to disrupt the operation of a victim VLM V by manipulating its output in a *targeted* manner, while evading a detector H_θ . To realize its objective, the attacker can perturb the visual modality of V . Concretely, it can apply a perturbation $p \in \mathcal{P}$ to the visual input $\mathbf{x}_i \in \mathcal{X}$ of the VLM through a transformation $\mathcal{A}$, resulting in an adversarial image $\mathcal{A}(\mathbf{x}_i, p)$. Aligned with the literature on optimization-based attacks (8; 18; 4; 21), we consider the transformation $\mathcal{A}$ adds the perturbation $p \in \mathcal{P}$ to $\mathbf{x}_i$ ensuring that the resulting adversarial image is within the valid pixel range, i.e., $\mathcal{A}(\mathbf{x}_i, p) \in \mathcal{X}$. The perturbation p could be constrained to an ϵ -ball for some ℓ_p norm for an adversarial perturbation (21), or it could be a physically realizable patch attack (27).

We consider a powerful attacker that has white-box access to V and H_θ , and can therefore utilize existing gradient-based adversarial attacks (8; 18; 4) to craft its attack. This allows us to stress-test adversarial defenses against the strongest attacker. Given an adversarial target output $t(\mathbf{x}_i) \neq V(\mathbf{x}_i, \mathbf{y}_i)$, the attacker’s objective is to find a perturbation $\hat{p}_i$ that, if applied to $\mathbf{x}_i$, results in the target output, i.e., $V(\mathcal{A}(\mathbf{x}_i, \hat{p}_i), \mathbf{y}_i) = t(\mathbf{x}_i)$, while remaining undetected, i.e., $H_\theta(\mathcal{A}(\mathbf{x}_i, \hat{p}_i), \mathbf{y}_i) = 0$.

Note that we assume that the attacker cannot manipulate the text input $\mathbf{y}_i$ (the user prompt). This assumption is reasonable in several practical applications such as perception (e.g., robotics, surveillance, autonomous driving) or spreading malicious images on social networks, where an attacker can control the visual modality but not the text, and the textual prompt is a predefined system feature.

4 *AdverStep*: Self-targeted Gradient Characterization

For a given image-text input $(\mathbf{x}_i, \mathbf{y}_i)$, we detect targeted visual adversarial attacks by leveraging the fact that adversarial inputs exhibit distinct optimization dynamics under iterative, self-targeted adversarial optimization. To capture these dynamics, we introduce the *adversarial gait*, a set of features that characterizes the evolution of loss and gradient signals during such test-time optimization. We then detect attacks in an unsupervised manner by comparing the adversarial gait to that of clean inputs and flagging significant deviations. Figure 2 provides a high-level overview of the approach.

The Adversarial Gait We compute the adversarial gait for an image-text input $(\mathbf{x}_i, \mathbf{y}_i)$ by performing a white-box self-targeted attack on the victim model V , whose attack target is given by $\mathbf{z}_i = V(\mathbf{x}_i, \mathbf{y}_i)$. That is, instead of misleading the model into producing an output that differs from $V(\mathbf{x}_i, \mathbf{y}_i)$, the self-targeted attack seeks to further increase the confidence in the predicted output. Similar to other white-box adversarial attacks, the self-targeted attack iteratively perturbs $\mathbf{x}_i$ using the gradient of the victim model’s cross entropy (CE) loss w.r.t. $\mathbf{x}_i$ at every iteration. Consider the

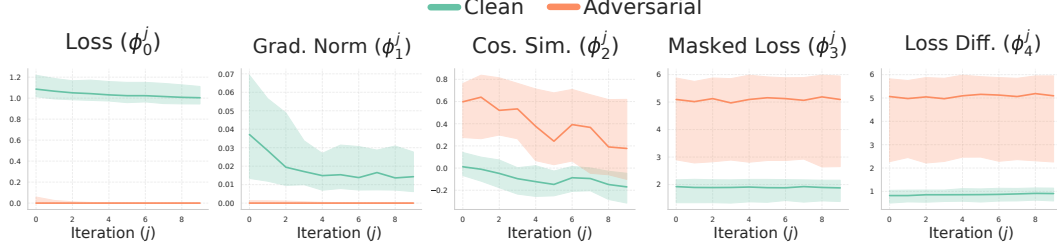


Figure 3: Evolution of adversarial gait features ϕ_i^j over multiple iterations j for clean and non-adaptive adversarial inputs. Solid lines and shaded regions represent the median and quartiles of the features’ data distribution, respectively.

adversarial objective loss function at iteration j of the self-targeted attack $\mathcal{L}(V, \mathbf{x}_i^j, \mathbf{y}_i, \mathbf{z}_i)$ and its gradient with respect to the input by $\nabla_{\mathbf{x}_i^j} \mathcal{L}(V, \mathbf{x}_i^j, \mathbf{y}_i, \mathbf{z}_i)$. The main intuition behind our proposed detector is that if $\mathbf{x}_i$ has been adversarially perturbed against V , this will have a noticeable impact on the self-targeted attack iterations in terms of loss and gradient. For simplicity, we denote said iteration loss and its gradient as $\mathcal{L}_j$ and ∇_j , respectively.

Hence, for a self-targeted attack consisting of N iterations, and N_f features constituting the adversarial gait, we define the adversarial gait $\mathbf{g} \in \mathbb{R}^{N_f \times N-1}$, where each column

$$\mathbf{g}_j = [\phi_0^j, \phi_1^j, \dots, \phi_{N_f-2}^j, \phi_{N_f-1}^j]^T, \quad (1)$$

contains N_f features, i.e., each $\phi_i^j \in \mathbf{g}_j$ is a real-valued feature computed in iteration j of the self-targeted attack. We propose to use $N_f = 5$ features that provide a thorough characterization of the adversarial loss and its gradient:

1. The first feature is the CE loss value, $\phi_0^j = \mathcal{L}_j$.
2. The second feature is the L2 norm of the gradient, $\phi_1^j = \|\nabla_j\|$. The first two features are motivated by the observation that the loss and the gradient norm of a self-targeted attack are noticeably lower when $\mathbf{x}_i$ has been subject to a targeted adversarial attack, as illustrated in Figure 3.
3. The third feature is the cosine similarity of consecutive gradients, i.e., $\phi_2^j = S_C(\nabla_j, \nabla_{j+1})$, where $S_C(\cdot)$ denotes the cosine similarity. The choice of this feature is motivated by the observation that the first few gradients computed on clean images exhibit a cosine similarity that is distinct from the one observed for images that have been subjected to an adversarial attack (see Figure 3).
4. To obtain the fourth feature, we compute the Hadamard product $B^j = \nabla_j \odot \nabla_{j+1}$ of the gradients at consecutive iterations, and we sum B^j across channels, yielding a two-dimensional matrix B_M^j . We then localize the pixels in $\mathbf{x}_i$ corresponding to the top p -th percentile of B_M^j , and replace them with gray pixel values (i.e., we set them to $128/255$), yielding the masked input $\mathbf{x}_i^{j*}$. Finally, we compute the loss function with respect to $\mathbf{x}_i^{j*}$, that is,

$$\mathcal{L}^{j*} = \mathcal{L}(V, \mathbf{x}_i^{j*}, \mathbf{y}_i, \mathbf{z}_i), \quad (2)$$

and we use this masked loss as the fourth feature $\phi_3^j = \mathcal{L}^{j*}$. Note that the target $\mathbf{z}_i$ is the predicted output based on the original unmasked input. This is because we are interested in quantifying the impact of occluding the regions that drive the current self-targeted attack iteration on the original objective. We find that the masked loss for adversarial images is clearly higher than for clean images, as illustrated in Figure 3.

5. The fifth and final feature is computed from the first and the fourth features as $\phi_4^j = \phi_3^j - \phi_0^j$. This feature captures the fact that the change in loss that results from masking, as shown in Figure 3, is larger for adversarial images.

In summary, the adversarial gait consists of the evolution of the above five features during a self-targeted attack: the self-targeted loss (ϕ_0^j), the gradient norm (ϕ_1^j), the cosine similarity between the

gradients of the current and the next self-targeted attack iterations (ϕ_2^j), the masked self-targeted loss (ϕ_3^j), and the difference between the masked and unmasked losses (ϕ_4^j). We illustrate how these features to differ between clean and adversarial features in Figure 3. In particular, the reason why we expect them to differ is that the self-targeted attack either (i) artificially maximizes the victim model’s confidence in its generated output for a clean image, or (ii) further maximizes the victim model’s confidence in the original attack target for an adversarial image. The intuition is that in order to successfully mislead the victim model, the attacker must approach a local optimum of (ii).

4.1 Unsupervised Detection based on the Adversarial Gait

In order to train a detector, we first consider a set $\mathcal{D}_{train}$ of clean image-text pairs, and compute the adversarial gait of each input $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_{train}$. As shown in Figure 3, most features do not change considerably across iterations j , and those that do (i.e., ϕ_1^j, ϕ_2^j), show a larger overlap for clean and adversarial inputs for higher j . Thus, we use a self-targeted attack with $N = 2$, i.e., we only compute the features ϕ_i^0 corresponding to $\mathbf{g}_0$. In the Appendix we explore the effect of using a higher N . In the following, we denote ϕ_i^0 by ϕ_i for simplicity. For each feature ϕ_i , we set two detection thresholds $\tau_i^{low}, \tau_i^{high} \in \mathbb{R}$ based on the values of ϕ_i derived from the images in $\mathcal{D}_{train}$.

During test-time, for input $(\mathbf{x}_i, \mathbf{y}_i)$ an attack is detected ($H_\theta(\mathbf{x}_i, \mathbf{y}_i) = 1$) if

$$\sum_{i=0}^{N_f-1} [\mathbb{1}_{\{\phi_i < \tau_i^{low}\}} + \mathbb{1}_{\{\phi_i > \tau_i^{high}\}}] > 0,$$

where $\mathbb{1}_{\{\cdot\}}$ is the indicator function. That is, an attack is detected if any feature ϕ_i is outside of its corresponding range $[\tau_i^{low}, \tau_i^{high}]$. Clearly, the choice of τ_i^{low} and τ_i^{high} represents a trade-off between false positives and the similarity that an adaptive attacker is required to achieve between the adversarial image’s gait and the gaits of clean images. We denote our approach as *AdverStep*.

5 Numerical Results

5.1 Evaluation Methodology

Datasets. We use 1000 image-text pairs from the VQA² validation split for training and tuning all defenses. For evaluation, we use four datasets for visual question answering (VQA) and image captioning tasks: NIPS2017 and VQA² for VQA, and MSCOCO and Flickr for image captioning. Due to the high computational cost of performing adaptive attacks, we limit our evaluation to 100 image-text pairs per dataset. To cope with the relatively small test set, we compute p-values and confidence intervals (CIs) using McNemar’s test and paired bootstrapping, respectively. Further details are available in the Appendix.

Victim Models and Baselines. We use SmolVLM-Instruct, Qwen2.5-VL-3B-Instruct, and Intern3-VL-2B as victim models. Unless otherwise stated, results correspond to Qwen-VL-3B. We compare *AdverStep* to recent detectors for visual adversarial attacks on VLMs, namely PIP (43), *Nearside* (13), and *MirrorCheck* (7); we omit *MirrorCheck* in some scenarios due to its high runtime and low performance (see the Appendix for additional baselines).

Defense and Attack Parameters. For *AdverStep*, to compute the *adversarial gait* we consider self-targeted attacks without budget constraints (besides ensuring valid pixel values), i.e., we use standard gradient descent, with a learning rate of $8/255^2$. To compute the masked loss, we use the top 5-th percentile of B_M^j . The Appendix contains all implementation details for *AdverStep* and the baselines. The detection thresholds for all defenses are based on our training set; they are set such that all defenses achieve the same false positive rate (FPR) on the training set. Unless otherwise specified, the FPR on clean images is set to 0%. Note that this is not the FPR on the evaluation dataset; hence we provide key results on the test-time FPR and further details on Appendix Sec. A.7. Importantly, the differences between training and test FPRs do not affect our main results.

For adversarial perturbations, we use PGD attacks with an ℓ_∞ norm bound of $16/255$ and a learning rate $\gamma = 1/255$, unless otherwise noted. We run the attacks for 1000 iterations. For patch attacks,

we use four square patches of equal size per image, without any constraints on their ℓ_∞ norm, and we use the same learning rate $\gamma = 1/255$. We always report the size of the region occupied by the patch attack as a percentage of the total image size, and it refers to the total area occupied by the four patches. See the Appendix for details on the attacks used for evaluation and those used for training our baselines.

Evaluation Metric. We use the attack success rate (ASR) to quantify the effectiveness of adaptive attacks. Given an adversarial test set $\mathcal{D}$, we define the ASR as the ratio of attacks that produce the target output while remaining undetected,

$$ASR = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}} \mathbb{1}_{\{V(\mathbf{x}_i, \mathbf{y}_i) = t(\mathbf{x}_i)\}} \cdot [1 - H_\theta(\mathbf{x}_i, \mathbf{y}_i)].$$

In Appendix A.9 we provide further details and additional evaluations using ASR_J , an alternative metric based on the LLM-as-a-judge framework (44).

Compute Hardware. Computations were performed on a server with a 64 core AMD EPYC 7A53 "Trento" CPU and four AMD MI250X GPU chips.

5.2 Detection of White-box Adaptive Adversarial Attacks

Table 1: ASR results for all datasets, models, and methods. **Bold:** statistically significant improvement over all baselines with a 95% CI and $p < 0.05$, an asterisk (*) denotes a 95% CI with $0.05 < p \leq 0.125$.

Model	NIPS				COCO			
	Ours	Nearside	PIP	MirrorCheck	Ours	Nearside	PIP	MirrorCheck
SmolVLM	0.91	1.00	1.00	1.00	0.480	1.00	1.00	1.000
Qwen-VL-3B	1.00	1.00	1.00	1.00	0.950	0.96	0.97	0.99
Intern-VL-2B	1.00	1.00	1.00	1.00	0.94*	0.98	0.98	0.98

Model	VQA ²				Flickr			
	Ours	Nearside	PIP	MirrorCheck	Ours	Nearside	PIP	MirrorCheck
SmolVLM	0.99	1.00	1.00	1.00	0.17	0.98	0.98	0.98
Qwen-VL-3B	0.68	0.86	0.85	0.88	0.82	0.94	0.86	0.95
Intern-VL-2B	0.81*	0.85	0.85	0.85	0.91*	0.96	0.960	0.96

Table 1 shows the performance of *AdverStep* and all baselines for adaptive attacks against different victim models and across different datasets, in terms of ASR. We note that *AdverStep* outperforms all baselines with a significant margin in most scenarios, and is not outperformed by any of the baselines in any scenario. The ASR is well above 0.5 for almost all combinations of defense, dataset, and victim model, which highlights the strength of our defense-aware threat model.

We observe no clear trends as to whether a particular dataset is more challenging or a specific model is more vulnerable, e.g., while *AdverStep*'s ASR for Qwen-VL-3B and Intern-VL-2B decrease when going from COCO to VQA², the opposite is true for SmolVLM. The NIPS dataset is particularly challenging for all defenses: all baselines are essentially unable to reduce the ASR, and *AdverStep* achieves a significant ASR reduction only for SmolVLM. However, it would be inaccurate to claim NIPS is the most challenging dataset since the ASR for *AdverStep* and SmolVLM is higher for VQA².

Table 2: ASR for various FPRs on NIPS and COCO. **Bold:** statistically significant improvement over all baselines (95% CI, $p < 0.05$). See Appendix Tab. 15 and Fig. 4 for details on test-time FPR.

Dataset	0% FPR			5% FPR			10% FPR			20% FPR		
	Ours	Ns	PIP	Ours	Ns	PIP	Ours	Ns	PIP	Ours	Ns	PIP
NIPS	1.00	1.00	1.00	0.53	1.00	0.96	0.39	1.00	0.97	0.24	1.00	0.95
COCO	0.95	0.96	0.97	0.21	0.97	0.95	0.17	0.90	0.95	0.13	0.84	0.95

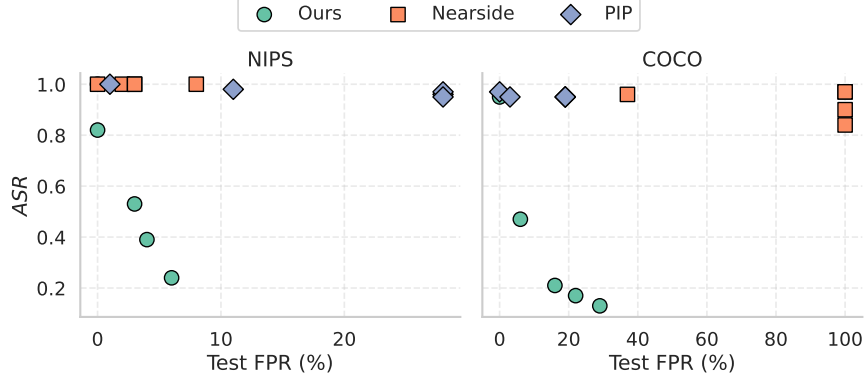


Figure 4: ASR vs. Test-time FPR. *AdverStep* achieves a much better trade-off than the baselines.

Impact of the FPR. For Table 1, the detection thresholds of all defenses were set to have a 0% FPR on the training data. In Table 2, we show the ASR for detection thresholds that result in other FPR values over the training data. For *AdverStep* we adjust τ_0^{low} and τ_4^{high} to allow for higher FPRs (see Appendix Sec. A.4 for details). The table shows that *AdverStep* achieves a favorable trade-off between ASR and FPR: allowing for an FPR of 5% yields a drop of roughly 50 and 75 percentage points on NIPS and COCO, respectively, and allowing for an FPR of 20% results in a drop that is around 75 and 80 percentage points, for NIPS and COCO, respectively. This reduction is remarkable, considering the baselines’ ASR, at any FPR for both datasets, drops by at most 12 percentage points.

Tuning all detectors to the same FPR on the training set enables a fair comparison where all of them set their detection thresholds without access to the evaluation set. However, this raises the question of what the resulting FPR is on the evaluation set, which is not captured by the ASR. In Figure 4 we show the ASR as a function of test-time FPR for *AdverStep*, *Nearside*, and *PIP*. We note that *AdverStep* attains a much lower ASR at a comparatively low test-time FPR: for NIPS the baselines cannot achieve the ASR that *AdverStep* has at 0% test-time FPR for any test-time FPR%. For COCO, the ASR of *AdverStep* at 0% test-time FPR is similar to that achieved by the baselines at any test-time FPR, and at a 6% test-time FPR it achieves an ASR vastly below the minimum ASR for the baselines at any FPR. Note also that *PIP* does not attain a test-time FPR of 0% on NIPS, while *Nearside*’s test-time FPR on COCO is well beyond 30% for any ASR.

Impact of the Attack Budget. So far we have shown results for perturbation attacks with an ℓ_∞ bound of 16/255. In Table 3 we show the ASR for various ℓ_∞ norm bounds, and for patch attacks of different sizes; all detectors are tuned to have 10% FPR on the training data.

Intuitively, larger ℓ_∞ bounds or patch sizes typically result in a higher ASR. For a bound of 8/255, the ASR experiences a sharp decrease compared to the default bound of 16/255 for all defenses, and in the case of *AdverStep*, it becomes nearly zero; note that *Nearside*’s ASR at this lower bound remains above that of *AdverStep* even for the original bound of 16/255. For 32/255 and 64/255 bounds, the ASR plateaus and the difference between the defenses diminishes, yet *AdverStep* still achieves a significantly lower ASR. A similar trend can be observed for patch attacks, with *AdverStep* attaining a significantly lower ASR for all budgets except for patches whose size is only 1% of the input image, where the ASR is nearly zero for all detectors.

AdverStep outperforms the baselines for a wide range of attack budgets, considering both norm-bounded and localized patch perturbations. Although a larger budget increases the ASR, it can incur considerable visual distortions in the case of norm-bounded attacks; in the case of patch attacks, using a large budget significantly increases the difficulty of deploying the attack in the physical world. Note that even at the largest budgets *AdverStep* significantly reduces the ASR.

Regarding the test-time FPR, which does not depend on the attack budget, we note that for the 10% training FPR used in our evaluation across attack budgets, *AdverStep*’s test-time FPR is at most three percentage points higher than either baseline (see Appendix Tab. 15): the tradeoffs illustrated in Figure 4 strongly suggest that a slight increase of less than three percentage points in test-time FPR, which would allow both baselines to attain or surpass *AdverStep*’s test-time FPR, would not suffice

for the baselines to bridge the gap between their ASR and that of *AdverStep* for attacker budgets where *AdverStep*'s ASR is significantly lower.

Table 3: ASR for various attack budgets (ℓ_∞ norm and patch size) for NIPS and COCO. **Bold**: statistically significant improvement over all baselines (95% CI, $p < 0.05$).

Norm-bounded attack					Patch attack			
Defense/Budget	8/255	16/255	32/255	64/255	1%	5%	10%	20%
<i>Nearside</i>	0.93	1.00	1.00	1.00	0.00	0.59	0.90	1.00
<i>PIP</i>	0.21	0.97	1.00	1.00	0.03	0.65	0.95	1.00
Ours	0.02	0.39	0.76	0.89	0.00	0.06	0.39	0.66

Norm-bounded attack					Patch attack			
Defense/Budget	8/255	16/255	32/255	64/255	1%	5%	10%	20%
<i>Nearside</i>	0.12	0.90	0.98	0.98	0.01	0.58	0.94	0.98
<i>PIP</i>	0.22	0.95	1.0	0.99	0.01	0.41	0.85	0.99
Ours	0.03	0.17	0.46	0.57	0.00	0.04	0.15	0.42

5.3 Computational Cost

In Table 4 we show, for each defense, the average time for a successful adaptive attack to be computed and the average computational cost of running the defense on a single image. Note that the averages are computed over all evaluation datasets. The table shows that *PIP* and *Nearside* are very computationally efficient and there is little difference between the two. *AdverStep* has a computational cost per input that roughly triples that of *PIP* and *Nearside*, but is still notably faster than *MirrorCheck*. The cost incurred by an adaptive attacker is much higher for *AdverStep*. A more detailed comparison for Qwen-VL-3B is presented in Figure 5. Although the time to success has a considerable spread, only a few outliers for *PIP* and *Nearside* are above the average for *AdverStep*. Similar figures for other victim models are available in the Appendix.

Importantly, except for *PIP* on Qwen-VL-3B, the ratio of each baseline’s time to success to that of *AdverStep* is lower than the ratio between that same baseline’s defense cost and *AdverStep*’s defense cost. Moreover, *AdverStep* consistently attains a lower ASR across datasets, attack budgets, and victim models, thus we conclude it offers a better tradeoff between computational cost and robustness.

We acknowledge that the overall latency of our defense is relatively high for safety-critical time-sensitive applications, but we argue that this is also true for baseline detectors, whose cost is in the same order of magnitude. Latency is widely considered an open challenge for time-sensitive applications of VLMs, such as autonomous driving, where even small architectures incur a cost of hundreds of milliseconds: the same order of magnitude as the cost for the detectors considered in our evaluation (11; 47).

Table 4: Average time to success of adaptive attacks and computational cost of all defenses. Results are in seconds. Best results per column are in **bold**.

Defense	SmolVLM		Qwen-VL-3B		Intern-VL-2B	
	Time to Succ.	Def. Cost	Time to Succ.	Def. Cost	Time to Succ.	Def. Cost
Nearside	21.808	0.382	100.211	0.596	27.093	0.312
PIP	23.849	0.460	146.667	0.644	33.021	0.351
MirrorCheck	20.513	10.415	59.282	10.636	26.906	10.432
Ours	70.849	1.087	421.150	1.893	133.006	0.996

5.4 Feature Ablation

To assess the impact of each feature ϕ_i that is used by *AdverStep* for detection, we perform an ablation study in which we remove each feature individually and report the resulting impact on the ASR. We report our results in Table 5, where we tune *AdverStep* to a 10% training FPR in all settings. Note that

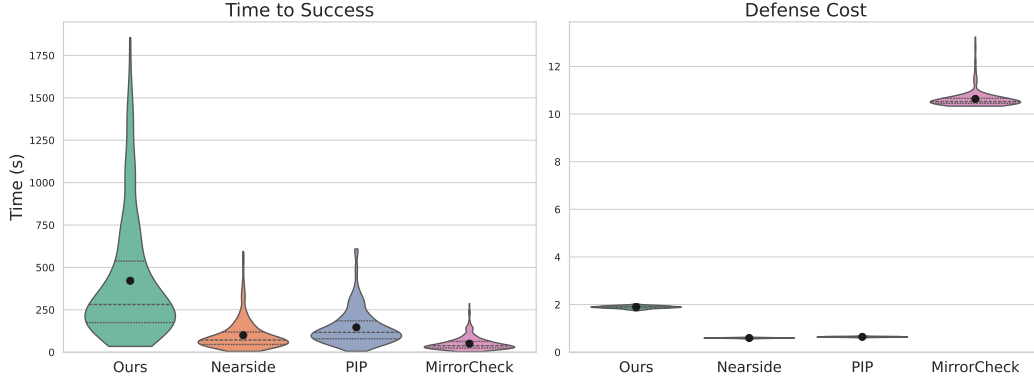


Figure 5: Time to success of adaptive attacks and computational cost of all defenses for Qwen-VL-3B. The violin plots show quartiles and the black dots represent the mean. *AdverStep* achieves a better trade-off than the baselines.

in order to sustain the 10% FPR in the absence of ϕ_0 , we lower τ_4^{high} , and similarly, in the absence of ϕ_4 we increase τ_0^{low} (details in Appendix Sec. A.4). Our results suggest that removing the gradient norm (ϕ_1) or cosine similarity (ϕ_2) has barely any impact on the ASR, while removing the loss (ϕ_0) or the masked loss (ϕ_3) results in a slight drop in ASR, which is only significant when removing ϕ_0 for NIPS. Most notably, removing the loss difference (ϕ_4) results in a very sharp increase in ASR.

By assessing the test-time FPR, we note that compared to the default setting, it is unchanged for $-\phi_1$, $-\phi_2$, and $-\phi_3$, it is slightly higher for $-\phi_0$, and drops to zero for $-\phi_4$ (see Tab. 16 in the Appendix). Hence, the impact on the ASR of removing a feature is mostly driven by the effect it has on the test-time FPR, which aligns with our results in Tab. 2. We note that the gradient norm and cosine similarity can be useful for adversarial detection, as shown in Figure 3 and as suggested in prior work (42; 12; 45), yet our results suggest they may not be particularly beneficial against adaptive attackers. We conjecture that the limited change for $-\phi_3$ is partly because ϕ_4 is obtained from ϕ_3 , and although the change is not statistically significant, the lower ASR obtained for $-\phi_3$ could be due to the masked loss providing useful information for the adaptive attacker through its BPDA approximation, and such information is not available when the adaptive loss only accounts for ϕ_4 .

Table 5: ASR for each *AdverStep* variant $-\phi_i$ where feature ϕ_i is excluded. The asterisk (*) denotes the original setting. Δ indicates change in ASR. **Bold**: significant change Δ (95%CI, $p < 0.05$).

Ablation	*	$-\phi_0$	$-\phi_1$	$-\phi_2$	$-\phi_3$	$-\phi_4$
ASR (NIPS)	0.39	0.27	0.39	0.4	0.32	0.95
Δ	0.0	-0.12	0.0	+0.01	-0.07	+0.56
ASR (COCO)	0.17	0.13	0.17	0.17	0.14	0.8
Δ	0.0	-0.04	0.0	0.0	-0.03	+0.63

6 Conclusion

In this paper, we propose *AdverStep*, a defense that leverages the gradient characteristics of self-targeted attacks for adversarial detection. Our defense trades off computational cost and false positives for robustness against adaptive attacks, and our results show that it outperforms existing baselines in a wide range of scenarios and victim models. We acknowledge that *AdverStep* incurs a computational overhead that is not suitable for time-critical applications, and, most importantly, that our evaluation is limited in scale and lacks an exhaustive coverage of adaptive attack formulations. Hence we outline efficiently leveraging the key components of *AdverStep* and exploring a more complete set of adversaries for evaluation as promising directions for future research.

We note that *AdverStep* displays remarkable gains over existing baselines in our evaluation centered around adaptive white-box targeted visual attack scenarios, but it is not a comprehensive solution for defense-aware threats to VLMs. We hope our work encourages a more thorough treatment of defense-aware threat models for future VLM adversarial defenses.

Acknowledgments

This work was partly funded by the KTH Railway Group, by the Swedish Transport Administration through Railway Excellence Area 6, and by the Swedish Research Council under grants 2024-05269 and 2025-06310. We acknowledge the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725, for computational and storage resources, and for awarding this project access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking and hosted by CSC (Finland) and the LUMI consortium.

References

- [1] Bryniarski, O., Hingun, N., Pachuca, P., Wang, V., Carlini, N.: Evading adversarial example detection defenses with orthogonal projected gradient descent. ArXiv **abs/2106.15023** (2021), <https://api.semanticscholar.org/CorpusID:235670132>
- [2] Byrd Victorica, M., Dán, G., Sandberg, H.: Saliutl: Ensemble salience guided recovery of adversarial patches against cnns. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2025)
- [3] Byrd Victorica, M., Dán, G., Sandberg, H.: SpaNN: Detecting multiple adversarial patches on CNNs by spanning saliency thresholds. In: Proc. of the IEEE Conference on Secure and Trustworthy Machine Learning (SaTML) (2025)
- [4] Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: IEEE Symposium on Security and Privacy (S&P). pp. 39–57 (2017)
- [5] Chen, L., Chen, Y., Luo, Y., Dou, H., Zhong, X.: Attention-guided hierarchical defense for multimodal attacks in vision-language models. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 1607–1617 (2025)
- [6] Croce, F., Goyal, S., Brunner, T., Shelhamer, E., Hein, M., Cemgil, T.: Evaluating the adversarial robustness of adaptive test-time defenses. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (eds.) Proceedings of the 39th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 162, pp. 4421–4435. PMLR (17–23 Jul 2022), <https://proceedings.mlr.press/v162/croce22a.html>
- [7] Fares, S., Ziu, K., Aremu, T., Durasov, N., Takáč, M., Fua, P., Nandakumar, K., Laptev, I.: Mirrorcheck: Efficient adversarial defense for vision-language models. arXiv preprint arXiv:2406.09250 (2024)
- [8] Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
- [9] Gou, Y., Chen, K., Liu, Z., Hong, L., Xu, H., Li, Z., Yeung, D.Y., Kwok, J.T., Zhang, Y.: Eyes closed, safety on: Protecting multimodal LLMs via image-to-text transformation. In: European Conference on Computer Vision. pp. 388–404. Springer (2024)
- [10] Hu, J., Zhou, J., Xie, M., Yang, Z., Li, J., Zheng, H., Shi, J., He, D., Li, Y.: Vlm-guard: Defending jailbreaks by monitoring only hundreds of safety-critical neurons. IEEE Transactions on Information Forensics and Security **21**, 4741–4754 (2026). <https://doi.org/10.1109/TIFS.2026.3690008>
- [11] Huang, J., Jin, Y.C., An, L., Park, J.: Litevlm: A low-latency vision-language model inference pipeline for resource-constrained environments. ArXiv **abs/2506.07416** (2025)
- [12] Huang, R., Geng, A., Li, Y.: On the importance of gradients for detecting distributional shifts in the wild. In: Neural Information Processing Systems (2021), <https://api.semanticscholar.org/CorpusID:238253184>
- [13] Huang, Y., Zhu, F., Tang, J., Zhou, P., Lei, W., Lv, J., seng Chua, T.: Effective and efficient adversarial detection for vision-language models via a single vector. ArXiv **abs/2410.22888** (2024), <https://api.semanticscholar.org/CorpusID:273695945>

- [14] Li, L., Guan, H., Qiu, J., Spratling, M.: One prompt word is enough to boost adversarial robustness for pre-trained vision-language models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 24408–24419 (2024)
- [15] Lin, Z., Zhao, Y., Chen, K., He, J.: I don’t know you, but I can catch you: Real-time defense against diverse adversarial patches for object detectors. In: Proc. of ACM Conference on Computer and Communications Security (CCS) (2024)
- [16] Liu, Q., Wang, F., Xiao, C., Chen, M.: Vlm-guard: Safeguarding vision-language models via fulfilling safety alignment gap. ArXiv **abs/2502.10486** (2025), <https://api.semanticscholar.org/CorpusID:276408250>
- [17] Lu, L., Pang, S., Liang, S., Zhu, H., Zeng, X., Liu, A., Liu, Y., Zhou, Y.: Adversarial training for multimodal large language models against jailbreak attacks. arXiv preprint arXiv:2503.04833 (2025)
- [18] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
- [19] Mao, C., Geng, S., Yang, J., Wang, X., Vondrick, C.: Understanding Zero-Shot Adversarial Robustness for Large-Scale Models. In: International Conference on Learning Representations (2023), <https://mlanthology.org/iclr/2023/mao2023iclr-understanding/>
- [20] Nie, W., Guo, B., Huang, Y., Xiao, C., Vahdat, A., Anandkumar, A.: Diffusion models for adversarial purification. In: International Conference on Machine Learning. pp. 16805–16827. PMLR (2022)
- [21] Qi, X., Huang, K., Panda, A., Wang, M., Mittal, P.: Visual adversarial examples jailbreak aligned large language models. In: AAAI Conference on Artificial Intelligence (2023), <https://api.semanticscholar.org/CorpusID:259244034>
- [22] Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PmLR (2021)
- [23] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10684–10695 (2022)
- [24] Rossolini, G., Nesti, F., Brau, F., Biondi, A., Buttazzo, G.: Defending from physically-realizable adversarial attacks through internal over-activation analysis. In: Proc. of the AAAI Conference on Artificial Intelligence (2023)
- [25] Schlarman, C., Singh, N.D., Croce, F., Hein, M.: Robust CLIP: Unsupervised adversarial fine-tuning of vision embeddings for robust large vision-language models. In: Proceedings of the 41st International Conference on Machine Learning (2024)
- [26] Seabold, S., Perktold, J.: statsmodels: Econometric and statistical modeling with python. In: 9th Python in Science Conference (2010)
- [27] Sun, J., Wang, C., Wang, J., Zhang, Y., Xiao, C.: Safeguarding vision-language models against patched visual prompt injectors. arXiv preprint arXiv:2405.10529 (2024)
- [28] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. CoRR **abs/1312.6199** (2013), <https://api.semanticscholar.org/CorpusID:604334>
- [29] Tarchoun, B., Khalifa, A.B., Mahjoub, M.A., Abu-Ghazaleh, N.B., Alouani, I.: Jedi: Entropy-based localization and removal of adversarial patches. In: Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2023)

- [30] Tramer, F., Carlini, N., Brendel, W., Madry, A.: On adaptive attacks to adversarial example defenses. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) *Advances in Neural Information Processing Systems*. vol. 33, pp. 1633–1645. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper_files/paper/2020/file/11f38f8ecd71867b42433548d1078e38-Paper.pdf
- [31] Tsiligkaridis, T., Roberts, J.: Understanding and increasing efficiency of frank-wolfe adversarial training. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 50–59 (2020), <https://api.semanticscholar.org/CorpusID:244728357>
- [32] Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* **17**, 261–272 (2020). <https://doi.org/10.1038/s41592-019-0686-2>
- [33] Wang, H., Wang, G., Zhang, H.: Steering away from harm: An adaptive approach to defending vision language model against jailbreaks. 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 29947–29957 (2024), <https://api.semanticscholar.org/CorpusID:274281170>
- [34] Wang, R., Ma, X., Zhou, H., Ji, C., Ye, G., Jiang, Y.G.: White-box multimodal jailbreaks against large vision-language models. In: *ACM MM*. p. 6920–6928 (2024). <https://doi.org/10.1145/3664647.3681092>
- [35] Wang, X., Chen, K., Zhang, J., Chen, J., Ma, X.: TAPT: Test-Time Adversarial Prompt Tuning for Robust Inference in Vision-Language Models. In: *Conference on Computer Vision and Pattern Recognition*. pp. 19910–19920 (2025). <https://doi.org/10.1109/CVPR52734.2025.01854>, <https://mlanthology.org/cvpr/2025/wang2025cvpr-tapt/>
- [36] Wang, Y., Ma, X., Bailey, J., Yi, J., Zhou, B., Gu, Q.: On the convergence and robustness of adversarial training. In: Chaudhuri, K., Salakhutdinov, R. (eds.) *Proceedings of the 36th International Conference on Machine Learning*. *Proceedings of Machine Learning Research*, vol. 97, pp. 6586–6595. PMLR (09–15 Jun 2019), <https://proceedings.mlr.press/v97/wang19i.html>
- [37] Wang, Z., Xie, C., Bartoldson, B., Kailkhura, B.: Double visual defense: Adversarial pre-training and instruction tuning for improving vision-language model robustness. *arXiv preprint arXiv:2501.09446* (2025)
- [38] Waseda, F., Tejero-de Pablos, A., Echizen, I.: Multimodal adversarial defense for vision-language models by leveraging one-to-many relationships. *arXiv preprint arXiv:2405.18770* (2024)
- [39] Xie, P., Bie, Y., Mao, J., Song, Y., Wang, Y., Chen, H., Chen, K.: Chain of attack: On the robustness of vision-language models against transfer-based adversarial attacks. In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. pp. 14679–14689 (2025)
- [40] Xu, Y., Qi, X., Qin, Z., Wang, W.: Cross-modality information check for detecting jailbreaking in multimodal large language models. In: *Conference on Empirical Methods in Natural Language Processing* (2024), <https://api.semanticscholar.org/CorpusID:271570997>
- [41] Zhang, J., Ye, J., Ma, X., Li, Y., Yang, Y., Chen, Y., Sang, J., Yeung, D.Y.: Anyattack: Towards large-scale self-supervised adversarial attacks on vision-language models. In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. pp. 19900–19909 (2025)
- [42] Zhang, S., Liu, F., Yang, J., Yang, Y., Li, C., Han, B., Tan, M.: Detecting adversarial data by probing multiple perturbations using expected perturbation score. *ArXiv abs/2305.16035* (2023), <https://api.semanticscholar.org/CorpusID:258887595>

- [43] Zhang, Y., Xie, R., Chen, J., Sun, X., Wang, Y.: Pip: Detecting adversarial examples in large vision-language models via attention patterns of irrelevant probe questions. In: Proceedings of the 32nd ACM International Conference on Multimedia. pp. 11175–11183 (2024)
- [44] Zheng, L., Chiang, W.L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J., Stoica, I.: Judging llm-as-a-judge with mt-bench and chatbot arena. In: Advances in Neural Information Processing Systems (2023)
- [45] Zheng, R., Dou, S., Zhou, Y., Liu, Q., Gui, T., Zhang, Q., Wei, Z., Huang, X., Zhang, M.: Detecting adversarial samples through sharpness of loss landscape. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Findings of the Association for Computational Linguistics: ACL 2023. Association for Computational Linguistics, Toronto, Canada (Jul 2023). <https://doi.org/10.18653/v1/2023.findings-acl.717>, <https://aclanthology.org/2023.findings-acl.717/>
- [46] Zhou, Q., Wang, D., Li, T., Lin, Y., Liu, Y., Dong, J.S., Guo, Q.: Defending LVLMs against vision attacks through partial-perception supervision. In: Forty-second International Conference on Machine Learning (ICML) (2025)
- [47] Zhou, X., Liu, M., Yurtsever, E., Zagar, B.L., Zimmer, W., Cao, H., Knoll, A.C.: Vision language models in autonomous driving: A survey and outlook. IEEE Transactions on Intelligent Vehicles (2024)

A Appendix

In the following we include additional resources, details, numerical results, and discussions to complement the content in the main body of our paper.

A.1 Code

The official code repository for our work is available at <https://github.com/gerkbyrd/AdverStep>.

A.2 Datasets

We use four datasets for evaluation, two consider a VQA task (NIPS2017 and VQA²), and two consider a captioning task (MSCOCO and Flickr). Note that NIPS2017 is a subset of ImageNet originally intended for targeted adversarial attacks on image classification. In our VQA setting, we use text descriptions for each ImageNet class and define the input prompt "Is this a [CLASS]?", similarly, for the adversarial target, we use the text descriptions and define the targeted attack objective as "No, it is a [TARGET CLASS]." The validation split of VQA² contains over 220K text-image pairs, where the text attached to each image is a question related to the image content (e.g., "What is the posted speed limit?"), for the adversarial target we merely use the ground truth answer corresponding to another image chosen at random, and no two images have the same adversarial target.

For the captioning task we use the test split from the MS-COCO 2014 captioning dataset where each image is paired with a text caption, and the test split of the Flickr30k dataset, where each image is paired with multiple captions. In these datasets, the adversarial target that we assign for each image is merely the caption corresponding to a different one, chosen at random. No two images have the same adversarial target.

Recall that for the VQA task, we only use 100 NIPS2017 and 100 VQA² images for evaluation. For the captioning task, we use 100 image subsets of COCO and Flickr. In the captioning task, all images use the same input query; in particular, throughout our evaluation we use the query "Please carefully observe the image and come up with a caption for the image." We use 1K VQA² images for training and tuning *AdverStep* and all baselines, which do not overlap with the 100 VQA² images used for evaluation.

A.3 Additional Baseline Defenses

Table 6: ASR at 10%FPR on NIPS including *Diffpure* and *Sharpness*.

	Diffpure	Sharpness	Nearside	PIP	Ours
ASR	1.0	1.0	1.0	0.97	0.39

In Section 5 we compared *AdverStep* to other detectors for visual adversarial attacks against VLMs. We now consider two relevant defenses that are not specifically designed for this setting. *Diffpure* (20) is representative of recovery methods. In short, it is a diffusion-based denoiser that is trained to remove adversarial perturbations from attacked images while maintaining the quality of clean images. *Sharpness* is an adversarial detector proposed by Zheng *et al.* (45). It is closely related to our work, since it performs an untargeted attack during test time to detect adversarial attacks on LLMs based on the loss function. We thus adapted it to our VLM visual attack scenario to compare it against *AdverStep*.

In Table 6 we show ASR results including these additional baselines on NIPS, where detectors are tuned to a 10%FPR on the training data. Note that *Diffpure* is a recovery method that attempts purification of the input image, hence it cannot be tuned according to a particular FPR on the training data. Our results show that similar to *PIP* and *Nearside*, *Sharpness* and *Diffpure* are vastly outperformed by *AdverStep*.

We acknowledge that *Sharpness* is originally proposed to detect attacks against LLMs, and an exhaustive tuning of its hyper-parameters is beyond the scope of our work. See Section A.4 for

further details on our implementation. We also note that while we cannot tune *Diffpure* to allow for a particular training FPR, its computational cost is already orders of magnitude greater than that of all other baselines, and the results of Nie *et al.* (20) suggest that attempting to reduce the ASR at the expense of larger computational overhead would present multiple difficulties. A larger diffusion timestep may indeed improve defensive performance, but the increase in computation time is significant compared to the cost of running *Diffpure* with its default parameters, which is quite high already. Moreover, the optimal timestep size reported by Nie *et al.* (20) is relatively small across threat models, and increasing it consistently degrades performance on clean images.

A.4 Victim Models, *AdverStep*, and Baselines Implementation Details

Victim Models. For all victim models, we limit the number of generated output tokens to 256 and use greedy sampling.

***AdverStep*.** In Sec. 5 we mention that we tune *AdverStep* for a given training FPR by manipulating τ_0^{low} and τ_4^{high} . Concretely, for a training FPR of 0%, we set all lower thresholds τ_i^{low} and higher thresholds τ_i^{high} according to the minimum and maximum values of their associated feature ϕ_i in the training data, respectively. Then, to allow for a higher training FPR of $f\%$, we set τ_0^{low} to the $\frac{f}{2}$ -th percentile for all values of ϕ_0 in the training data, and we set τ_4^{high} to the $(100 - \frac{f}{2})$ -th percentile for the values of ϕ_4 in the training data. Note that this actually sets a lower and an upper bound of $\frac{f}{2}\%$ and $f\%$, respectively, on the training FPR.

When considering the ablation settings $-\phi_0$ and $-\phi_4$ presented in Table 5 setting a training FPR of $f\%$ is more straightforward. In the case of $-\phi_0$, we set τ_4^{high} to the $(100 - f)$ -th percentile of the values of ϕ_4 in the training data. Conversely, for $-\phi_4$, we set τ_0^{low} to the f -th percentile of the values of ϕ_0 in the training data.

Baselines. Our implementations for all baselines are available in our code repository. We implement *PIP* (43) following the details from the original work regarding SVM parameters, generation of adversarial training data, and computation of attention patterns; the only difference is that we use our proposed training dataset instead of the specific ImageNet partition used by Zhang *et al.* (43), and for generation of adversarial attacks we use a learning rate of $\gamma = 1$. For *Nearside* (13), we compute the attacking direction using the hidden state corresponding to the last input token as in the original work; instead of generating adversarial data using 5000 steps, we use 1000 steps. For *MirrorCheck* we use Stable Diffusion XL-B-1.0 as the text-to-image generator model and CLIP ViT-B-16 to compute the text-image embedding similarity. Since *Diffpure* does not require setting any detection threshold, we directly use the official implementation (20).

In order to adapt *Sharpness* (45) for our visual attack setting, we apply the untargeted adversarial perturbations on pixel space instead of embedding space; we noticed on preliminary experiments that perturbing the image in embedding space led to very poor performance even for non-adaptive attacks. We use 4 steps and the adaptive step size as in the official implementation, with a base learning rate of $4/255$; we also set $g_{min} = 0.1$ which is the default value. We set the norm bound of the untargeted attack to $4/255$, and use the ℓ_2 norm following the original work. We use our training data to tune the detection threshold using the FPR; *Sharpness* does not require any reference adversarial data to choose a detection threshold.

A.5 Statistical Significance and Confidence Intervals

In this section we describe how we use McNemar’s test and paired bootstrapping to compute p-values and confidence intervals, and report detailed CIs and p-values for the results reported in Sec. 5.

We perform McNemar’s test of homogeneity using the binomial distribution, where our reported p-value corresponds to the null hypothesis of equal marginal distributions; we use the implementation of the statsmodels library (26). To obtain confidence intervals to assess the difference in ASR between *AdverStep* and each of the baselines, we compute the difference of ASR over all inputs for each evaluation set, and then we perform paired bootstrapping using $10K$ resamples. We rely on the SciPy (32) bootstrap function and use the bias-corrected and accelerated bootstrap confidence interval; we always compute a 95% confidence interval.

Table 7: Detailed CIs and p-values for Table 1. The CIs are in percentage points. A dash (-) indicates [0,0] p=1.

Dataset	Model	<i>Nearside</i>	<i>PIP</i>	<i>MirrorCheck</i>
NIPS	SmolVLM	$[-16, -4] p = .0039$	$[-16, -4] p = .0039$	$[-16, -4] p = .0039$
	Qwen-VL-3B	-	-	-
	Intern-VL-2B	-	-	-
COCO	SmolVLM	$[-62, -42] p < .0001$	$[-62, -42] p < .0001$	$[-62, -42] p < .0001$
	Qwen-VL-3B	$[-6, 3] p = 1$	$[-7, 1] p = .625$	$[-10, -1] p = .125$
	Intern-VL-2B	$[-10, -1] p = .125$	$[-10, -1] p = .125$	$[-10, -1] p = .125$
VQA ²	SmolVLM	$[-5, 0] p = 1$	$[-5, 0] p = 1$	$[-5, 0] p = 1$
	Qwen-VL-3B	$[-26, -11] p < .0001$	$[-25, -10] p < .0001$	$[-28, -13] p < .0001$
	Intern-VL-2B	$[-9, -1] p = .125$	$[-9, -1] p = .125$	$[-9, -1] p = .125$
Flickr	SmolVLM	$[-88, -72] p < .0001$	$[-88, -72] p < .0001$	$[-88, -72] p < .0001$
	Qwen-VL-3B	$[-20, -6] p = .0018$	$[-11, 1] p = .3438$	$[-21, -8] p = .0002$
	Intern-VL-2B	$[-10, -2] p = .0625$	$[-10, -2] p = .0625$	$[-10, -2] p = .0625$

Table 8: Detailed CIs and p-values for Table 2. The CIs are in percentage points. A dash (-) indicates [0,0] p=1.

NIPS				
	0%FPR	5%FPR	10%FPR	20%FPR
<i>Nearside</i>	-	$[-57, -37] p < .0001$	$[-70, -51] p < .0001$	$[-84, -67] p < .0001$
<i>PIP</i>	-	$[-53, -33] p < .0001$	$[-67, -48] p < .0001$	$[-79, -62] p < .0001$

COCO				
	0%FPR	5%FPR	10%FPR	20%FPR
<i>Nearside</i>	$[-6, 3] p = 1$	$[-83, -67] p < .0001$	$[-81, -63] p < .0001$	$[-79, -61] p < .0001$
<i>PIP</i>	$[-7, 1] p = .625$	$[-82, -64] p < .0001$	$[-85, -68] p < .0001$	$[-88, -73] p < .0001$

For Tables 1, 2, and 3, we show the resulting confidence intervals and p-values in Tables 7, 8, and 9, respectively. In the main results, we claim that *AdverStep*'s ASR is lower with a 95% CI if the upper value of the corresponding CI for ASR (*AdverStep*) - ASR (baseline) is negative. For Table 5, we show CIs for the change Δ and the corresponding p-values in Table 10; we claim that Δ is a significant change (either decrement or increment) when its CI does not contain zero and it has $p < 0.05$.

A.6 On the Adaptive Attack Evaluation

Our adaptive defense-aware threat model is at the core of our main findings. Nonetheless, we acknowledge that our results are not based on an exhaustive coverage of possible adaptive attack formulations. Therefore, in this section we first provide a thorough description of our adaptive attack formulation, and then we present additional ablation studies on the adaptive attack algorithm used for evaluation. The purpose of this section is to validate that our main results are not highly dependent on particular details of our adaptive attack implementation.

Adaptive Attack Details The computation of adaptive attacks depends on the defense scheme as well as on the detection threshold.

For *AdverStep*, the loss of the adaptive attacker can be written as

$$\mathcal{L}_{adapt} = \max(\phi_0 - \tau_0^{low}, 0) + \sum_{i=0}^{N_f-1} [\max(\phi_i - \tau_i^{high}, 0) + \max(\tau_i^{low} - \phi_i, 0)].$$

Recall that ϕ_0 represents the cross-entropy loss for the self-targeted attack. From the attacker's perspective, this is also the loss for the adversarial objective of misleading the victim model V , because

Table 9: Detailed CIs and p-values for Table 3. The CIs are in percentage points. Recall norm-bounded attacks correspond to ℓ_∞ budgets in $\{8/255, 16/255, 32/255, 64/255\}$, and patch attacks to area budgets in $\{1\%, 5\%, 10\%, 20\%\}$.

Dataset	Budget	<i>Nearside</i>	<i>PIP</i>
NIPS	8/255	$[-96, -84] p < .0001$	$[-28, -12] p < .0001$
	16/255	$[-70, -51] p < .0001$	$[-67, -48] p < .0001$
	32/255	$[-33, -16] p < .0001$	$[-33, -16] p < .0001$
	64/255	$[-18, -6] p = .001$	$[-18, -6] p = .001$
	1%	$[0, 0] p = 1$	$[-8, -1] p = .25$
	5%	$[-64, -42] p < .0001$	$[-69, -50] p < .0001$
	10%	$[-65, -45] p < .0001$	$[-65, -45] p < .0001$
	20%	$[-43, -25] p < .0001$	$[-43, -25] p < .0001$
COCO	8/255	$[-15.4, -4] p = .0039$	$[-28, -12] p < .0001$
	16/255	$[-81, -63] p < .0001$	$[-85, -68] p < .0001$
	32/255	$[-62, -42] p < .0001$	$[-64, -44] p < .0001$
	64/255	$[-51, -32] p < .0001$	$[-52, -33] p < .0001$
	1%	$[-5, 0] p = 1$	$[-6, 0] p = 1$
	5%	$[-63, -43] p < .0001$	$[-47, -28] p < .0001$
	10%	$[-86, -70] p < .0001$	$[-78, -60] p < .0001$
	20%	$[-66, -46] p < .0001$	$[-66, -47] p < .0001$

if the attacker successfully misleads V , then the output of V , which is the target of *AdverStep*'s self-targeted attack, will correspond to the attacker's target. Therefore, the first term in $\mathcal{L}_{adapt}$ ensures the attack optimizes for the adversarial objective as long as the loss does not fall beneath its lower detection threshold. The second term ensures that the attack optimizes for detection features to fall within the normal range defined by their corresponding thresholds. $\mathcal{L}_{adapt}$ is just a sum of clipped versions of the features ϕ_i , hence its gradient w.r.t. the features can be computed directly. However, ensuring that the gradient computation can flow from all features to the adversarial input image is not trivial.

The gradients of the cross-entropy loss (ϕ_0) w.r.t the attacker's perturbation can be computed directly. Since the gradient norm (ϕ_1), the cosine similarity (ϕ_2), the masked loss (ϕ_3), and the loss difference (ϕ_4), are computed using gradients from consecutive self-targeted attack iterations, we compute second order gradients to ensure these features are differentiable with respect to the input image when we add them to the loss function. Note that the masked loss (ϕ_3) and hence the loss difference (ϕ_4) involve non-differentiable operations, namely the computation of the masking threshold and the masking itself. To allow backpropagation through the masked regions, we rely on the Backward Pass Differentiable Approximation (BPDA) framework for the backward pass, approximating the binary mask with a sigmoid function with a sharpness parameter set to 10; we detach the masking threshold for the gradient computation.

For *Nearside*, *PIP*, and *Sharpness* the detection score can be added to the adversarial loss directly, as it is a differentiable function of the victim model's hidden states for *Nearside*, a function of the victim model's attention patterns for *PIP*, and the loss corresponding to the test-time untargeted attack for *Sharpness*. Hence for these baselines we use

$$\mathcal{L}_{adapt-b} = \min(\det(\mathbf{x}_i) - \tau_b, 0) + \mathcal{L}_{CE}.$$

Where $\det(\mathbf{x}_i)$ represents the detection score, and τ_b the detection threshold.

For *MirrorCheck* we opt for a simpler approach than that proposed by Fares *et al.* (7), and instead of using the full detection pipeline, we first generate samples from the text-to-image generator

Table 10: Detailed CIs and p-values for the change Δ in Table 5. The CIs are in percentage points. A dash (-) indicates $[0,0] p=1$.

Dataset	$-\phi_0$	$-\phi_1$	$-\phi_2$	$-\phi_3$	$-\phi_4$
NIPS	$[-21, -3] p < .0169$	-	$[-2, 5] p = 1$	$[-17, 2] p < .21$	$[44, 65] p < .0001$
COCO	$[-11, 1] p = .2891$	-	-	$[-8, 1] p < .3750$	$[52, 73] p < .0001$

Table 11: ASR of adaptive attacks against *AdverStep* as a function of PGD attack iterations.

Iters.	0	100	200	300	400	500	600	700	800	900	1K
NIPS	0	0.43	0.88	0.96	0.99	0.99	0.99	0.99	1.00	1.00	1.00
COCO	0	0.20	0.55	0.71	0.82	0.84	0.88	0.91	0.94	0.95	0.95
VQA	0	0.26	0.37	0.47	0.56	0.61	0.65	0.65	0.68	0.68	0.68
Flickr	0	0.08	0.30	0.47	0.56	0.63	0.70	0.74	0.77	0.80	0.82

used for detection, based on the attack target; then during each attack iteration we compute the embedding similarity of the adversarial image and the pre-computed generated images, and compare the minimum similarity across samples to the detection threshold τ_{mc} to formulate the adaptive loss, that is

$$\mathcal{L}_{adapt-mc} = \min(\tau_{mc} - \min_{x_j^{gen} \in \mathcal{X}_{gen}} [\text{sim}(\mathbf{x}_i, x_j^{gen})], 0) + \mathcal{L}_{CE},$$

where τ_{mc} is *MirrorCheck*'s detection threshold (an attack is detected when the similarity is below τ_{mc}), and $\mathcal{X}_{gen}$ is the set of images generated by *MirrorCheck*'s text-to-image model; in our experiments, sampling a single image from the text-to-image model was sufficient. Thus, the adaptive attacker aims to maximize the embedding similarity between the adversarial image and the images generated using the adversarial target.

For *Diffpure*, we leverage the approach proposed by Nie *et al.* (20) to compute adaptive attacks against *Diffpure* in a memory-efficient manner, and thus we can compute the gradient of the adversarial loss w.r.t. the purified input image and the gradient of the purified image w.r.t. the original input image. For all baselines, the attacker knows the detection thresholds, hence during iterations where the features used for detection are able to evade the detector, their contribution to the total loss is zero; note that this does not apply to *Diffpure*.

We rely on well established approaches to approximate the gradient of the non-differentiable operations involved in *AdverStep*. However, this does not guarantee that the reported gains over the baselines are not the result of gradient-masking. We argue that the non-negligible ASR against *AdverStep* in most scenarios, and its increase for larger attack budgets, suggest that our adaptive attack is indeed able to optimize against *AdverStep*'s detection pipeline. In the following, we provide additional results to validate the adaptive attacks against *AdverStep* used in our evaluation.

ASR vs. Adaptive Attack Iterations As an initial sanity check, we assess the ASR of adaptive attacks against *AdverStep* as a function of the PGD iterations. We show the results for Qwen-VL-3B in Table 11. We note that the adaptive attack's success rate increases with the number of attack iterations, plateauing as the iterations go to the fixed limit of 1000. The consistent relationship between ASR and iterations indicates our adaptive attack effectively optimizes for the intended objective by using the proposed algorithm and loss function. Additionally, it suggests the gains of further increasing the number of attack steps would be negligible and offer a poor tradeoff between attack effectiveness and computational cost for the attacker.

Non-adaptive attacks In Figure 1 we show that *AdverStep* has a stronger performance on non-adaptive attacks, and the increase in ASR for the adaptive case suggests the attack used for evaluation is indeed able to optimize against *AdverStep*. The results in Figure 1 correspond to Qwen-VL-3B on COCO, where all detectors were tuned at a 10%FPR on the training set. For completeness, in Table 12 we present the ASR for non-adaptive attacks against Qwen-VL-3B on NIPS and COCO, across different FPRs on the training data. The non-adaptive attacks use the same optimization parameters as our adaptive attacks described in Sec. 5.1, but only aim to minimize the attack objective loss (i.e., they only aim to mislead the victim model). Due to the reduced cost of non-adaptive attacks, we use 500 test images from each dataset.

The results confirm *AdverStep*'s strong performance for non-adaptive attacks. In particular, at a 0%FPR it achieves a near-zero ASR, notably below that of the baselines, and achieves an ASR of zero for a 5%FPR, which *Nearside* can only attain for NIPS at a 20%FPR and *PIP* cannot attain at any FPR on either dataset. Apart from demonstrating *AdverStep*'s suitability for non-adaptive attacks, these results are a useful sanity check that confirms optimizing for our proposed adaptive loss, with

the same hyper-parameters and algorithm, enables the attacker to leverage gradient-descent to evade *AdverStep*.

Table 12: ASR for various FPRs on NIPS and COCO for non-adaptive attacks. The undefended ASR is 1.0 for NIPS and 0.96 for COCO.

Dataset	0% FPR			5% FPR			10% FPR			20% FPR		
	Ours	Ns	PIP	Ours	Ns	PIP	Ours	Ns	PIP	Ours	Ns	PIP
NIPS	0.002	0.13	0.036	0.00	0.002	0.006	0.00	0.002	0.006	0.00	0.00	0.006
COCO	0.014	0.252	0.08	0.00	0.008	0.012	0.00	0.004	0.012	0.00	0.002	0.012

Sensitivity to BPDA Masking Approximation A key step in *AdverStep*’s detection pipeline is the computation of the binary mask B_M^j , which is used to compute ϕ_3 and ϕ_4 . By default we set the sharpness of the sigmoid function used to approximate the mask to 10. To test the sensitivity of our proposed adaptive attack against *AdverStep* to this approximation, we conduct an ablation on the sharpness parameter for Qwen-VL-3B on NIPS, and present our results in Table 13; note that *AdverStep* is tuned to a 10%FPR on the training set. Our results show a negligible decrease in ASR for lower sharpness values, and suggest that our proposed adaptive attack has a low sensitivity to this parameter. We emphasize this does not mean parameters ϕ_3 and ϕ_4 are not important for detection, in fact Table 5 reflects the importance of our masking-based features for detection.

Alternative Attack Algorithms As described in Sec. 5.1, we perform PGD for 1000 iterations with a learning rate of $\gamma = 1/255$. While our main results involve a wide range of attack budgets, we do not modify the optimization algorithm used by the attacker. Defenses against adversarial attacks are usually evaluated against different attack algorithms targeting the victim model.

Note that *AdverStep* performs unsupervised detection, i.e., its detection thresholds are set exclusively using clean images, hence regardless of the attacker’s optimization algorithm, the attacks used for evaluation are always unseen attacks. However, it has been established that a faulty adaptive attack implementation can easily overestimate the robustness of adversarial defenses (30).

For completeness, we assess the ASR achieved by the adaptive attacker using two alternative attack algorithms. The first one is the Carlini-Wagner attack (4), where the attacker has an unlimited attack budget, but includes the ℓ_2 norm of the applied perturbation in its loss function, weighed by a scalar λ , which we set to 0.01. The second one is Orthogonal PGD (OPGD) (1), which is particularly insightful as it is specifically designed to achieve effective adaptive attacks against adversarial detectors. We present results for these attacks and compare them to the default PGD attack with an ℓ_∞ norm of $16/255$ in Table 14.

The results show that *AdverStep* and the baselines achieve a much lower ASR for the CW attack, whose ASR is only above 5% for *Nearside* on the NIPS dataset: at least without further tuning, the CW attack is not well suited for our adaptive attack scenario compared to our default PGD attack. We also note that the perturbations found by the Carlini-Wagner attack against *AdverStep* have ℓ_∞ norms that range roughly between $22/255$ and $64/255$, confirming that the higher ASR achieved by the PGD attacker is not due to the CW attacker disproportionately reducing the attack budget.

The OPGD attack achieves a substantially higher ASR against *AdverStep*, but it is still well below than the ASR OPGD reaches for the baselines. Interestingly, the ASR drops against *Nearside* on the COCO dataset when using OPGD, which indicates that it is not always preferable to our proposed adaptive attack. Moreover, we point out OPGD requires checking attack success at every iteration, which results in an increased computational overhead for the attacker; since we optimize jointly in our approach, the attacker can check for the success condition only during iterations where the adaptive attack evades the detector, whereas OPGD requires checking said condition (which requires generating the victim model’s output) for every iteration. We consider that a more thorough evaluation using an ensemble of adaptive attackers, and assessing each detector based on its worst-case ASR across the ensemble, would be a valuable direction for future work.

Table 13: ASR for different values of the sigmoid sharpness.

Sig. Sharpness	2	5	10 (default)	20
ASR	0.38	0.38	0.39	0.39

Table 14: Average ASR for *AdverStep* and all baselines.

Model	NIPS			COCO		
	Ours	<i>Nearside</i>	<i>PIP</i>	Ours	<i>Nearside</i>	<i>PIP</i>
PGD (default)	0.39	1.00	0.97	0.17	0.90	0.95
CW	0.03	0.47	0.04	0.0	0.05	0.01
OPGD	0.74	1.00	0.98	0.44	0.75	0.97

A.7 On Test Time FPR

Recall that for our results, we tune *AdverStep* and all detectors to have a certain FPR on the training dataset (usually 0% or 10%). Our rationale is to ensure a fair comparison where each detector is tuned on the same calibration data, without using the evaluation data at all. We show the tradeoff between test-time FPR and ASR in Table 2, which partially addresses the question of how much the test-time FPR diverges from the training time FPR. To cover said question more thoroughly, we now present detailed results for the test-time FPR on all evaluation datasets in Table 15.

We observe that *AdverStep* achieves the lowest test-time FPR at most training FPRs across datasets, and when either baseline has a lower test-time FPR, the difference is relatively small. Note that all detectors are better calibrated for some datasets than others. *Nearside* has a fairly low test-time FPR for NIPS and VQA², which is always below its training FPR, however, it suffers from a very high FPR for COCO and Flickr, which is essentially 100% for all training FPRs above zero. *PIP*’s test-time FPR rapidly surpasses its training FPR when the latter is above zero, but it usually plateaus when the training FPR is above 5%, with a maximum value of roughly 30% for VQA²: interestingly, the training set is a non-overlapping subset from the same VQA² dataset, indicating that it is not trivial to generalize to the unseen VQA² test set with access to our training data. *AdverStep* achieves test-time FPRs below their corresponding training FPRs on NIPS and VQA², it also achieves a test-time equal to or below that of the baselines on COCO for 0% and 5% training FPRs, and on Flickr for all training FPRs below 20%.

Table 15: Test-time FPRs (in percentage points) across training-time FPRs.

Method	NIPS					COCO				
	0%	1%	5%	10%	20%	0%	1%	5%	10%	20%
<i>Nearside</i>	0	2	3	3	8	37	100	100	100	100
<i>PIP</i>	1	11	28	28	28	0	3	19	19	19
Ours	0	0	3	4	6	0	6	16	22	29

Method	VQA ²					Flickr				
	0%	1%	5%	10%	20%	0%	1%	5%	10%	20%
<i>Nearside</i>	0	1	3	8	18	17	99	100	100	100
<i>PIP</i>	1	13	30	30	31	1	5	26	26	26
Ours	0	0	2	3	9	1	2	9	19	29

Overall, in certain instances our results do indicate that test-time FPRs are notably higher than the training FPRs, but we argue this does not impact our main findings in Sec. 5. Regarding Tab. 1, and any results involving a 0% training FPR for that matter, we note that *AdverStep* has a test-time FPR that is equal or lower to that of the baselines. For results that consider a 10% training FPR, we concede that *AdverStep*’s test-time FPR is one percentage point higher than that of *Nearside* for NIPS, and three percentage points higher than that of *PIP* for COCO; but as we discuss in Sec. 5, Table 15 shows that in spite of this *AdverStep* still has a much better tradeoff between ASR and test-time FPR.

Table 16: Test-time FPR for all ablations of *AdverStep* with a training FPR of 10%. The asterisk (*) denotes the original setting. $-\phi_i$ denotes the case where feature ϕ_i is excluded. Δ indicates absolute change in Test-time FPR.

Ablation	*	$-\phi_0$	$-\phi_1$	$-\phi_2$	$-\phi_3$	$-\phi_4$
Test FPR (NIPS)	4	6	4	4	4	0
Δ	0	+2	0	0	0	-4
Test FPR (COCO)	22	29	22	22	22	0
Δ	0	+7	0	0	0	-22

Table 17: ASR for cross-calibration experiment on NIPS and COCO. All detectors are tuned for a 10% training FPR; the corresponding test-time FPR is shown in brackets.

Dataset	Ours	<i>Nearside</i>	<i>PIP</i>
NIPS (tuned on COCO)	0.0 [8]	1.0 [0]	0.99 [24]
COCO (tuned on NIPS)	0.06 [33]	0.90 [100]	0.96 [1]

Feature Ablation and Test-time FPR For the feature ablation results in Table 5, we mention that the impact of removing any one feature used for detection on the ASR is related to the impact this has on the detection threshold and thus the test-time FPR. In Table 16 we report the test-time FPRs for all ablations of *AdverStep* presented in Table 5, when the training FPR is set to 10%.

Effect of Training Dataset We obtain our training set from VQA² due to its diversity in terms of both images and input prompts. To investigate the effect of using a different dataset for training and tuning of *AdverStep* and our baseline detectors, we perform a cross-calibration experiment using our NIPS and COCO evaluation sets. We train and/or tune each detector on NIPS and evaluate on COCO and viceversa, always using a training FPR of 10%. We report our results in Table 17.

For NIPS, tuning on COCO instead of VQA² yields looser detection thresholds for the baselines, evidenced by the slight drop in test-time FPR. This explains why *Nearside* still has an ASR of 1.0 and *PIP* experiences a negligible increase in ASR (c.f. Table 2). In contrast, tuning on NIPS doubles *AdverStep*’s test-time FPR, but its ASR drops to zero. For COCO, when tuning on NIPS instead of VQA², *Nearside*’s test-time FPR remains unchanged at 100%, and its ASR also remains unchanged w.r.t. Table 2. For *PIP*, although test-time FPR notably drops to nearly zero, the ASR only sees a negligible increase of one percentage point. Once again *AdverStep* sees a considerable increase in its test-time FPR, which is congruent with the large drop in ASR.

Overall, the results are in line with those in Table 2 and Figure 4: the impact on the ASR of calibrating on a different dataset is ultimately driven by how tight the resulting detection threshold is, as evidenced by the correspondence between test-time FPR and ASR.

A potential advantage of tuning the detectors on the evaluation distribution would be a reduced divergence between training and test-time FPR, but we consider the difference between calibration and test sets to be a strength of our evaluation.

A.8 Overview of Adversarial Defenses for VLMs

Detection. The literature for VLM adversarial input detection is quickly expanding, but remains relatively underexplored when compared to recovery and adversarial training approaches. *PIP* (43) uses attention weights from the victim VLM as the input to a support vector machine (SVM) classifier. *Nearside* (13) extracts an *attack direction* by computing the difference between adversarial and benign inputs in the embedding space of the victim VLM. The projection of the input’s embedding onto the attack direction is used as a detection score at test-time. *VLM-Guard* (10) is a similar approach that computes reference "safety-critical" vectors based on benign and adversarial activations of a set of "safety-critical" neurons; said neurons are monitored at test-time to compute any given input’s safety-critical vector, whose cosine similarity to the reference vectors is leveraged for adversarial detection.

MirrorCheck (7) generates an image from the text output of the victim VLM (using a diffusion model (23)) and compares its semantic similarity to the input image (using a multimodal embedding model (22)) in order to reveal inconsistencies introduced by the attack. *CIDER* (40) computes the embedding similarity between the image and the text in the input using an external model before and after purifying the image with a diffusion-based denoiser. The difference between the similarity before and after purification is used as an adversarial detection score. These last two approaches are costly due to the latency introduced by the diffusion models.

Recovery. Many recent approaches to recovery steer away from supervised learning to avoid reliance on labeled adversarial examples, potentially improving generalization. *SmoothVLM* (27) is specifically proposed against patch attacks. It performs pixel-masking of random locations across different versions of the same image and aggregates the VLM responses from all versions. *ECSO* (9) converts images into text before feeding them into VLMs in order to leverage the relative robustness and safety alignment of language-only models. *APT* (14) optimizes the embeddings corresponding to one suffix token to be added to the prompt to increase the robustness of VLMs to adversarial attacks; *TAPT* (35) also relies on so-called defensive prompts for recovery, but instead of learning a fixed suffix it proposes to tune the prompt during test-time by minimizing an unsupervised loss that captures prediction consistency across augmentations and embedding alignment between regular and adversarially-optimized prompts. *DPS* (46) generates image descriptions based on multiple partial perceptions of the potentially malicious image, and uses them to aid and supervise the final prediction by the VLM.

Adversarial training. Adversarial training relies on re-training or fine-tuning the model using examples of benign and adversarial images in order to enhance robustness. For instance, Lu *et al.* (17) propose adversarial training for VLMs against jail-breaking attacks, while avoiding large overheads by concentrating the fine-tuning on added "visual projection" modules. Focusing on image-text interactions, Waseda *et al.* (38) propose multi-modal adversarial training incorporating adversarial perturbations to both the image and text modalities, and showed that this significantly outperforms uni-modal approaches. Recently, Chen *et al.* (5) performed attention-guided multi-modal adversarial training aiming to align the attention distributions of adversarial and clean images. Furthermore, Wang *et al.* (37) proposed a two-phased approach for VLM adversarial training, where a CLIP vision encoder is adversarially pre-trained and then used within a VLM that is also adversarially trained end-to-end. Mao *et al.* (19) propose a text-guided contrastive adversarial loss (TeCoA), which can be leveraged to fine-tune a VLM to enhance its adversarial robustness, or alternatively perform visual prompt-tuning, akin to some recovery approaches discussed above (14; 35). Schlarmann *et al.* (25) propose an unsupervised fine-tuning approach, which defines a training loss based on the distance between perturbed and clean embeddings. The results reported in these works indicate that the proposed techniques considerably boost adversarial robustness when compared to vanilla adversarial training.

A.9 On ASR-based Evaluation for Detection

Detection methods are usually evaluated using the trade-off between false and true positive rates (e.g., via ROC curves, precision, recall, F1-score, etc.). We choose to instead evaluate based on the ASR because in an adaptive attack scenario the detection threshold should be known when creating adversarial images. Hence our evaluation quantifies how often an adaptive attacker is able to evade the detector and mislead the victim model, for a given detection threshold. We can still assess how the detectors' performance relates to the FPR, as we do in Figure 4, but achieving the same level of granularity typically used in ROC curves would be computationally very expensive, since the adaptive attacks must be created from scratch for each detection threshold.

For our evaluation and threat model, we define a successful targeted attack as one where the victim's output is the attack target, i.e., $V(\mathbf{x}_i, \mathbf{y}_i) = t(\mathbf{x}_i)$. While this definition is straightforward for other tasks (e.g., image classification) in tasks like automatic captioning and visual-question answering, a more concrete definition is required. Firstly, our attack targets are specific textual answers, hence we must clarify that we consider $V(\mathbf{x}_i, \mathbf{y}_i)$ to represent the VLMs generated output with a temperature of zero. Second, an exact equality of the target and output texts can be unreasonably strict, e.g., the attack may be considered unsuccessful if it is off by a punctuation mark or the capitalization (or lack thereof) of a single letter. To partially cope with this issue, we always consider an attack successful

if the victim’s output equals the target after both have been stripped of all punctuation marks and capitalizations.

LLM as a Judge We acknowledge our definition of attack success is still quite strict, but we consider it is more suitable for our evaluation compared to other approaches. For example, the LLM as a Judge (LaaJ) framework (44) can be useful to determine whether an attack has been successful based on semantic similarity between the victim’s output and the target, however, evaluating said similarity during the adaptive attack to check the success condition would be prohibitively costly. Similarly, relying on embedding similarity between the victim’s output and the attack target could be an efficient alternative; attack success based on a threshold for the similarity is indeed a less strict, semantically-driven definition of success, however, preliminary experiments suggest it may be too loose, in that texts with notable discrepancies may be deemed sufficiently similar according to such a threshold.

We now present our main results considering an ASR metric based on the LaaJ framework, which we denote ASR_J . Following Xie *et al.* (39), we define it as

$$ASR_J = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}} JUDGE(o_i, V(\mathbf{x}_i, \mathbf{y}_i), t(\mathbf{x}_i)) \cdot [1 - H_\theta(\mathbf{x}_i, \mathbf{y}_i)],$$

where o_i represents the clean output of the model and $JUDGE(.) \in \{0, 0.5, 1.0\}$ denotes an LLM-based score, which is zero when $V(\mathbf{x}_i, \mathbf{y}_i)$ is similar to the clean output, one when it is similar to the attack target, and 0.5 when it is not similar to either. We use Qwen2.5-14B-Instruct to implement the $JUDGE(.)$ function, using greedy decoding and an output token limit of 512. We also use the evaluation prompt proposed by Xie *et al.* (39):

Below is the original text, the target text, and the generated text of a multimodal model. Please judge whether the victim model has been attacked successfully. Think through this problem step by step. - If the generated text is similar to the original text, assign a score of 0, indicating the attack failed.- If the generated text is unrelated to both the target text and the original text, assign a score of 0.5, indicating the model is fooled but has not yet produced the target text.- If the generated text is roughly similar to the actual generated text and the main objects are consistent, assign a score of 1, indicating the targeted attack was completely successful. Additionally, display the score in a separate line from the final analysis process.

Original Text: [Original Text]

Target Text: [Target Text]

Generated Text: [Generated Text]

As noted before, using the success definition during the adaptive attack optimization is too costly, hence we leverage the adversarial images created using our default definition of success. For instances where the adaptive attack is successful according to our strict success definition, we assume the score of the Judge is one, and for cases where it is not successful, we compute the score for the adversarial image corresponding to the attack iteration with the lowest adaptive loss, considering only iterations where the detector in turn was evaded.

Table 18 shows the ASR_J for all baselines on all datasets, it is analogous to the ASR results in Table 1. By definition, the ASR_J is equal or higher than the ASR, which is confirmed when comparing these two tables. *AdverStep* achieves a noticeably lower ASR_J compared to the baselines for SmolVLM on NIPS, COCO, and Flickr, and for Qwen-VL-3B on VQA². On all other scenarios, the difference in ASR_J between all methods becomes negligible. While *AdverStep* still has the upper hand over the baselines, under the LaaJ-based notion of attack success its advantage over other detectors fades for most scenarios.

Table 19 shows the ASR_J for *AdverStep* and the baselines across different training FPRs. This table is analogous to the ASR results in Table 2. Although the ASR_J is considerably higher than the ASR, the results confirm that *AdverStep* still outperforms the baselines for FPR values above 0%. *AdverStep* is able to tradeoff a higher FPR for a lower ASR_J , while the baselines’ ASR_J is very close to one at any FPR.

Table 18: ASR_j for all datasets, models, and methods.

Model	NIPS				COCO			
	Ours	Nearside	PIP	MirrorCheck	Ours	Nearside	PIP	MirrorCheck
SmolVLM	0.940	1.000	1.000	1.000	0.890	1.000	1.000	1.000
Qwen-VL-3B	1.000	1.000	1.000	1.000	0.980	0.975	0.985	1.0
Intern-VL-2B	1.000	1.000	1.000	1.000	0.997	0.995	0.995	0.995

Model	VQA ²				Flickr			
	Ours	Nearside	PIP	MirrorCheck	Ours	Nearside	PIP	MirrorCheck
SmolVLM	0.995	1.000	1.000	1.000	0.790	1.000	1.000	1.000
Qwen-VL-3B	0.805	0.910	0.900	0.910	0.980	1.000	0.995	1.000
Intern-VL-2B	0.880	0.880	0.905	0.900	0.985	0.995	1.000	0.995

Table 19: ASR_j for various FPRs on the NIPS and the COCO datasets.

Dataset	0% FPR			5% FPR			10% FPR			20% FPR		
	Ours	Ns	PIP	Ours	Ns	PIP	Ours	Ns	PIP	Ours	Ns	PIP
NIPS	1.000	1.000	1.000	0.840	1.000	0.985	0.695	1.000	0.985	0.730	1.000	0.990
COCO	0.980	0.975	0.985	0.875	1.000	0.995	0.870	0.950	0.970	0.850	0.980	0.995

Table 20: ASR_j for different attack budgets for NIPS and COCO datasets.

Norm-bounded attack					NIPS				Patch attack			
Defense/Budget	8/255	16/255	32/255	64/255	1%	5%	10%	20%	1%	5%	10%	20%
<i>Nearside</i>	0.985	1.000	1.000	1.000	0.235	0.890	1.000	1.000	0.235	0.890	1.000	1.000
<i>PIP</i>	0.775	0.985	1.000	1.000	0.205	0.900	1.000	1.000	0.205	0.900	1.000	1.000
Ours	0.515	0.695	0.930	0.975	0.335	0.495	0.695	0.950	0.335	0.495	0.695	0.950

Norm-bounded attack					COCO				Patch attack			
Defense/Budget	8/255	16/255	32/255	64/255	1%	5%	10%	20%	1%	5%	10%	20%
<i>Nearside</i>	0.780	0.950	0.995	0.990	0.355	0.895	0.990	0.995	0.355	0.895	0.990	0.995
<i>PIP</i>	0.845	0.970	1.000	0.995	0.180	0.723	0.965	0.995	0.180	0.723	0.965	0.995
Ours	0.735	0.870	0.950	0.970	0.470	0.560	0.700	0.880	0.470	0.560	0.700	0.880

Finally, Table 20 shows the ASR_j for *AdverStep* and the baselines across different attacker budgets. This table is analogous to the ASR results in Table 3. As expected, the ASR_j in Table 20 is higher than the ASR in the corresponding scenarios in Table 3. Overall, *AdverStep* still displays a stronger robustness to increases in the attacker budget, but in certain scenarios the difference to the baselines becomes quite small, specifically norm-bounded budgets above 16/255 on both datasets and a 20% patch budget for NIPS. Interestingly, for the 1% patch attack budget the baselines outperform *AdverStep*, although it has a clearly lower ASR_j for patch attacks with larger budgets, for which the baselines' ASR_j rises very sharply.

Overall, the LaaJ definition of success is more lenient for the adaptive attacker, yielding a notable performance reduction for all detectors in most scenarios. Importantly, even under this wider definition, we find that *AdverStep* still achieves notable reductions of ASR_j and in general outperforms the baselines across our evaluation scenarios. We argue that the most adequate definition of attack success depends on the intended deployment scenario, LaaJ may be preferable to achieve, e.g., a misleading automatic caption, whereas a strict definition may be preferable for targeted attacks where semantic similarity is insufficient, e.g., generating links to malicious websites.

A.10 Additional Computational Cost Results

In Figure 5 we showed a detailed comparison of the cost of bypassing *AdverStep* and all baselines, as well as the cost of running each defense, for the Qwen-VL-3B victim model. In Figure 6 and Figure 7 we show similar results for SmolVLM and Intern-VL-2B, respectively. The same tendency

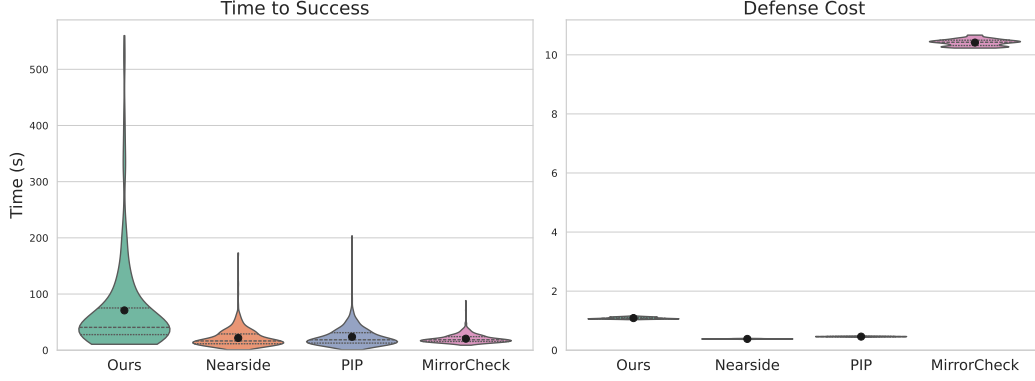


Figure 6: Time to success of adaptive attacks and computational cost of all defenses for SmolVLM. The violin plots show quartiles and the black dots represent the mean. *AdverStep* achieves a better trade-off than the baselines.

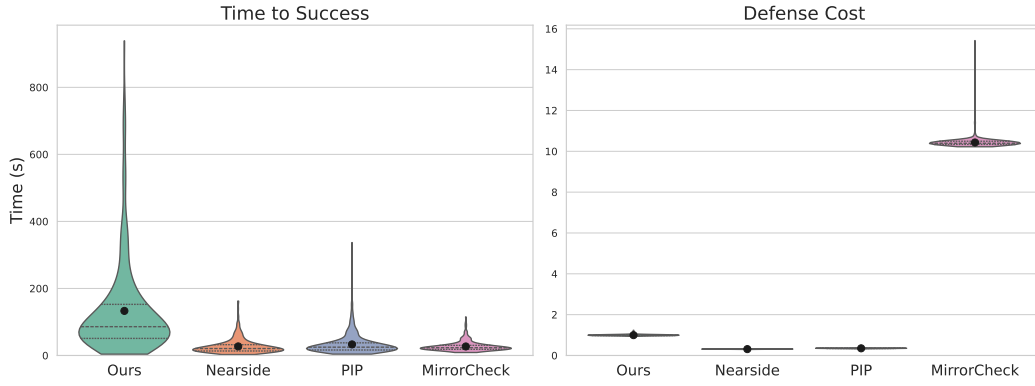


Figure 7: Time to success of adaptive attacks and computational cost of all defenses for Intern-VL-2B. The violin plots show quartiles and the black dots represent the mean. *AdverStep* achieves a better trade-off than the baselines.

observed for Qwen-VL-3B is present for these victim models: the adaptive attacker incurs a notably higher cost to bypass *AdverStep* than it does to bypass the baselines. The defense cost results are also congruent with Table 4 and our analysis in Section 5.

Time Metrics. Our results in Table 4, Figure 5, Figure 6, and Figure 7, are based on two metrics: the time to success and defense cost. The defense cost is merely the time it takes to execute a given defense. The time to success is based on the ASR metric, that is,

$$\text{Time to Success} = T_0 + \sum_{j_{adv}=1}^{j_{adv}^*} T_{j_{adv}},$$

where j_{adv} denotes the adaptive attack iteration, $T_{j_{adv}}$ denotes computational cost at iteration j_{adv} , and j_{adv}^* denotes the iteration at which the strict definition of success used to compute the ASR is achieved. That is, the adversarial image $\mathbf{x}_i^{j_{adv}^*}$ produced at iteration j_{adv}^* satisfies $V(\mathbf{x}_i^{j_{adv}^*}, \mathbf{y}_i) = t(\mathbf{x}_i)$ and $H_\theta(\mathbf{x}_i^{j_{adv}^*}, \mathbf{y}_i) = 0$. T_0 is outside of the summation to account for one-time costs incurred by the adaptive attacker, such as the image generation step in *MirrorCheck*. Note that this metric is defined only for successful attacks.

Table 21: ASR for different self-targeted attack steps N for *AdverStep* on NIPS tuned at a 10% training FPR. Test-time FPR is in brackets. The asterisk (*) denotes the original setting. Δ indicates change in ASR.

N	2*	3	4	5
ASR	0.39 [4]	0.42 [4]	0.45 [4]	0.45 [4]
Δ	0.0	+0.03	+0.06	+0.06

Table 22: ASR at different self-targeted learning rates for *AdverStep* at 10% training FPR. Test-time FPR is in brackets. The asterisk (*) denotes the original setting. Δ indicates change in ASR w.r.t. default setting. **Bold**: significant change Δ (95%CI, $p < 0.05$).

Learning Rate	1/255 ²	8/255 ² *	16/255 ²
ASR (NIPS)	0.27 [4]	0.39 [4]	0.20 [3]
Δ	-0.12	0.0	-0.19
ASR (COCO)	0.14 [19]	0.17 [22]	0.09 [21]
Δ	-0.03	0.0	-0.08

A.11 Additional Ablation Studies

In Table 5 we reported our results when ablating each adversarial gait feature used for detection. We now explore ablations of other key components in *AdverStep*. First, we perform detection with an adversarial gait that is based on a self-targeted attack with a larger number of iterations N than our default setting of $N = 2$. That is, on top of the features we used for detection in our evaluations, we also leverage the same set of features corresponding to further steps of the self-targeted attack.

In Table 21, we show the results on NIPS for the impact on the ASR and test-time FPR when the self-targeted attack is performed for higher values of N . In short, while the change is not statistically significant, we observe that initially the ASR increases for higher N , but the ASR stops increasing at $N = 4$; the test-time FPR remains unchanged. In Figure 3, we showed that increasing the number of iterations used to compute the gait is not particularly beneficial to distinguish clean images from non-adaptive adversarial images. The results in Table 21 are in line with this intuition, since the observed change in ASR is not significant. However, we note the effect of a higher N is reminiscent of the effect removing the masked loss ϕ_3 in Table 5: both result in a change of ASR without affecting the test-time FPR.

We conjecture that similar to the inclusion of ϕ_3 , using a higher N may not benefit our detector, but may provide useful information to the adaptive attacker, leading to the slight increase in ASR. Note that increasing the N iterations represents a linear increase in computational cost for *AdverStep* and hence also for the adaptive attacker. We consider that exploring alternative aggregation schemes across self-targeted iterations, in order to effectively leverage a higher N for detection, and further investigating how the attacker can benefit from using a higher N than *AdverStep* when crafting its attacks, are interesting directions for future work.

By default our self-targeted attack used for detection has a learning rate of 8/255², we obtain results for NIPS and COCO using different learning rates for the self-targeted attack, and report our results in Table 22. Interestingly, we find that reducing or increasing the learning rate of the self-targeted attack can significantly reduce the ASR, and we find no instances where it increases the ASR in our evaluation. Moreover, in contrast to the feature ablations in Table 5, the variants in Table 22 lead to a similar or even slightly lower test-time FPR, indicating that the reduction in ASR does not come at the expense of a notable increase in FPR.

We conclude that tuning the self-targeted learning rate instead of fixing it to our default value of 8/255² could enhance the performance of *AdverStep* in terms of the tradeoff between ASR and FPR. We consider that a more exhaustive assessment that incorporates other attack budgets and formulations (e.g., patch attacks, alternative algorithms) is a promising direction for future work.

Untargeted Gradient Characterization Our proposed detector considers targeted attacks, which relate to diverse relevant real-world scenarios for VLMs. The targeted objective of our threat model

Table 23: ASR of *AdverStep* and *AdverStep-u* on NIPS and COCO.

Dataset	<i>AdverStep</i>	<i>AdverStep-u</i>
NIPS	0.39	0.94
COCO	0.17	0.72

is key to our design, and as we discuss in Sec. 4, it is the reason why we propose to use a self-targeted attack to compute the adversarial gait instead of using an alternative attack objective.

However, a natural question to ask is whether our approach could potentially benefit from an alternative attack during test-time to compute the gait features. We test an alternative version of *AdverStep*, denoted *AdverStep-u*, which performs an untargeted attack instead of a self-targeted attack to compute the adversarial gait used for detection; we tune this detector to a 10%FPR on the training data and evaluate it for adaptive attacks against Qwen-VL-3B on NIPS and COCO. We report our results in Table 23. Our results show a sharp increase in ASR, even though we use the same features to characterize the gait, and the same adaptive loss, it becomes easier for the adaptive attacker to evade *AdverStep-u* while misleading the victim model. This relates to the main intuition behind our self-targeted attack: there is no obvious reason why the gait for an untargeted attack that minimizes the victim’s output for a clean input, should notably differ from the gait corresponding to an untargeted attack that minimizes the victim’s output for a successful adversarial input.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope?

Answer: [Yes]

Justification: All main claims are concretely addressed by our numerical results and analyses.

Guidelines:

- The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: We discuss limitations pertaining to our threat model and proposed detection scheme.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate “Limitations” section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren’t acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: No theoretical results are presented in our paper.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We describe our proposed detection method and attack algorithms used for evaluation.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We use open-source models and provide an anonymized repository with our proposed detection method.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://neurips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://neurips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We provide and justify our design choice in the main paper and the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Where appropriate, we present our results using summary statistics such as the median and inner quartiles.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide details of our hardware resources.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: Our work complies with the NeurIPS guidelines.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Safety-critical scenarios of ML applications are used to motivate our work.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Our code repository does not involve such high risk components.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We include references to existing assets involved in our paper.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[Yes\]](#)

Justification: The code for our proposed detection method is available as an anonymized repository.

Guidelines:

- The answer [\[N/A\]](#) means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [\[N/A\]](#)

Justification: This paper does not involve human subjects.

Guidelines:

- The answer [\[N/A\]](#) means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [\[N/A\]](#)

Justification: Our paper does not involve human subjects.

Guidelines:

- The answer [\[N/A\]](#) means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs have not been used to develop the methodology presented in this paper.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.