MirrorCheck: Efficient Adversarial Defense for Vision-Language Models

Supplementary Material

A. Further Background and Related Works

A.1. Visual-Language Models (VLMs)

Humans possess the remarkable ability to seamlessly integrate information from various sources concurrently. For instance, in conversations, we adeptly interpret verbal cues, body language, facial expressions, and intonation. Similarly, VLMs demonstrate proficiency in processing such multimodal signals, allowing machines to comprehend and generate image-related content that seamlessly merges visual and textual components. Contemporary VLM architectures such as CLIP [55] predominantly leverage transformer-based models [62, 63] for processing both images and text due to their effectiveness in capturing long-range dependencies. At the heart of the transformers lies the multi-head attention mechanism, which plays a pivotal role in these models' functionality.

To enable multimodal comprehension, VLMs typically comprise three key components: *(i)* an Image Model responsible for extracting meaningful visual features from visual data, *(ii)* a Text Model designed to process natural language, and *(iii)* a Fusion Mechanism to integrate representations from both modalities. Encoders in VLMs can be categorized based on their fusion mechanisms into Fusion encoders [64–67], which directly combine image and text embeddings, Dual encoders [2, 3, 55, 68], which process modalities separately before interaction, and Hybrid methods [69, 70] that leverage both approaches. Furthermore, fusion schemes for cross-modal interaction can be classified into single-stream [64, 66, 67, 69, 70] and dual-stream [65] architectures. The recent surge in multimodal development, driven by advances in vision-language pretraining (VLP) methods, has led to diverse vision-language applications falling into three main categories: *(i)* Image-text tasks (such as image captioning, retrieval, and visual question answering), *(ii)* Core computer vision tasks (including image classification, object detection, and image segmentation), and *(iii)* Video-text tasks (such as video captioning, video-text retrieval, and video question-answering).

A.2. Other Adversarial Attacks used against VLMs

Attack-Bard [24]. For a victim model that is a Multimodal Large Language Model (MLLM), adversarial examples that effectively perturb the image embeddings of Bard [71] will consequently impact the text generation process. Let x represent a natural image and $\tilde{\mathcal{I}}_{i\phi}()$ be a set of surrogate image encoders. The image embedding attack is defined as solving the following optimization problem:

$$\text{argmax}_{\delta:\|\delta\|_\infty \leq \varepsilon} \sum_{i=1}^{N} \|\tilde{\mathcal{I}}_{i\phi}(x_{adv}) - \tilde{\mathcal{I}}_{i\phi(x)}\|_2^2 \tag{12}$$

where $x_{adv} = x + \delta$ and the goal is to maximize the difference between the embeddings of the adversarial image x_{adv} and the natural image x while ensuring that the perturbation δ remains within a specified threshold ϵ. To address the optimization problem in (12), [24] employed the SSA-CWA approach, as introduced in [72].

Attack-MMFM [48]. An untargeted attack proposed against multimodal foundation models. To introduce minor perturbations to the visual inputs of a VLM, the authors propose a white-box untargeted attack. Specifically, given a natural image x, a ground truth caption t, along with context images c and context text z, the objective is to design an attack that increases the negative log-likelihood of the target text t^* within the constraints of the threat model:

$$\max_{\delta_x, \delta_c} -\sum_{i=1}^{m} \log p(t_i^* \mid t_{<i}^*, z, x + \delta_x, c + \delta_c) \tag{13}$$

$$\text{s.t. } \|\delta_x\|_\infty \leq \epsilon_x, \|\delta_c\|_\infty \leq \epsilon_c$$

In equation 13 above, δ_x is the perturbation to the input image and δ_c is the perturbation to the context images. In the setting where only the input images are attacked, optimization is performed only on δ_x and $\epsilon_c = 0$.

A.3. Adversarial Attacks used for Classification

An adversarial example, within the scope of machine learning, is a sample intentionally manipulated by an adversary to provoke an incorrect output from a target classifier. Typically, in image classification tasks, where the ground truth is based on human perception, defining adversarial examples involves perturbing a correctly classified sample (referred to as the seed example) by a limited amount to generate a misclassified sample (denoted as x_{adv}). Existing research on adversarial example generation predominantly centers on image classification models, reflecting the prominence and vulnerability of such models to adversarial attacks. Numerous methodologies have been introduced to craft adversarial examples, encompassing fast gradient-based techniques [20, 73], optimization-based strategies [22, 74], and other innovative approaches [75, 76]. Notably, [22] introduced state-of-the-art attacks that impose constraints on L_0, L_2, and L_∞ norms, highlighting the versatility and effectiveness of adversarial attacks across various norm spaces.

Adversarial examples can be categorized as targeted or untargeted depending on the adversary's objective. In targeted attacks, the adversary aims for the perturbed sample x_{adv} to be classified as a specific class, while in untargeted attacks, the objective is for x_{adv} to be classified as any class other than its correct class.

Formally, a targeted adversary seeks to find an x_{adv} such that the target classifier assigns it to the target class y while remaining within a certain distance ϵ from the original sample x_{clean}. Conversely, an untargeted adversary aims to find an x_{adv} which is misclassified compared to the original x_{clean} within the same distance threshold ϵ. The adversary's strength, denoted as ϵ, restricts the allowable transformations applied to the seed example. In contrast, the distance metric $\Delta(x_{\text{clean}}, x_{\text{adv}})$ and the threshold ϵ model how close an adversarial example needs to be to the original to deceive a human observer. As specified in our related work section, we will introduce some attack strategies used in classification tasks. We also leverage these attacks to test the efficacy of `MirrorCheck` in this setting;

- **Fast Gradient Sign Method** (FGSM, L_∞, Untargeted): The Fast Gradient Sign Method (FGSM) is an adversarial attack technique proposed by Goodfellow et al. [20] that efficiently generates adversarial examples for deep neural networks (DNNs). The objective of the FGSM attack is to perturb input data in such a way that it induces misclassification by the target model while ensuring the perturbations are imperceptible to human observers. The main idea behind FGSM is to compute the gradient of the loss function with respect to the input data, and then perturb the input data in the direction that maximizes the loss. Specifically, FGSM calculates the gradient of the loss function with respect to the input data, and then scales the gradient by a small constant ϵ to determine the perturbation direction. This perturbation is added to the original input data to create the adversarial example. Mathematically, the FGSM perturbation is defined as:

$$x_{\text{adv}} = x_{\text{clean}} + \epsilon \cdot \text{sign}(\nabla_x J(w^T x_{\text{clean}}), y))$$

  where ϵ is a small constant controlling the magnitude of the perturbation, and sign denotes the sign function. The objective function of the FGSM attack is typically the cross-entropy loss between the predicted and true labels, as it aims to maximize the model's prediction error for the given input.
- **Basic Iterative Method** (BIM, L_∞, Untargeted): The Basic Iterative Method (BIM) attack [77], also known as the Iterative Fast Gradient Sign Method (IFGSM), is an iterative variant of the FGSM attack designed to generate stronger adversarial examples. Like FGSM, the objective of the BIM attack is to craft adversarial perturbations that lead to misclassification by the target model while remaining imperceptible to human observers. In the BIM attack, instead of generating a single perturbation in one step, multiple small perturbations are iteratively applied to the input data. This iterative approach allows for finer control over the perturbation process, resulting in adversarial examples that are more effective and harder for the target model to defend against. The BIM attack starts with the original input data and applies small perturbations in the direction of the gradient of the loss function with respect to the input data. After each iteration, the perturbed input data is clipped to ensure it remains within a small ϵ-ball around the original input. This process is repeated for a fixed number of iterations or until a stopping criterion is met. Mathematically, the perturbed input at each iteration s of the BIM attack is given by:

$$x^s_{\text{adv}} = \text{clip}_\epsilon(x^{s-1}_{\text{adv}} + \alpha \cdot \text{sign}(\nabla_x J(w^T x_{\text{clean}}), y))$$

  where Clip_ϵ denotes element-wise clipping to ensure the perturbation magnitude does not exceed ϵ, and α is a small step size controlling the magnitude of each perturbation. The BIM attack aims to maximize the loss function while ensuring the perturbations remain bounded within the ϵ-ball around the original input.
- **DeepFool** (L_2, Untargeted): The DeepFool attack [51] is an iterative and computationally efficient method for crafting adversarial examples. It operates by iteratively perturbing an input image in a direction that minimally changes the model's prediction. The objective of the DeepFool attack is to find the smallest perturbation that causes a misclassification while ensuring that the adversarial example remains close to the original input in terms of the L_2-norm. The DeepFool attack starts with the original input image and iteratively computes the perturbation required to push the image across the decision boundary of the model. It computes the gradient of the decision function with respect to the input and then finds the direction in which the decision boundary moves the most. By iteratively applying small perturbations in this direction, the DeepFool attack gradually moves the input image towards the decision boundary until it crosses it. Mathematically, the perturbed input at each iteration of the DeepFool attack is computed as follows:

$$x^s_{\text{adv}} = x^{s-1}_{\text{adv}} + \alpha \cdot \frac{\nabla_{\text{f}}(x_{\text{clean}})}{\|\nabla_{\text{f}}(x_{\text{clean}})\|_2}$$

  where x^{s-1}_{adv} is the input image at the current iteration s, α is a small step size, and $\nabla_{\text{f}}(x)$ is the gradient of the decision function with respect to the input image x_{clean}. The process continues until the model misclassifies the perturbed input or until a maximum number of iterations is reached.
- **Projected Gradient Descent** (PGD, L_2, Untargeted): The Projected Gradient Descent (PGD) attack [13] is

an advanced iterative method used for crafting adversarial examples. It builds upon the Basic Iterative Method (BIM), extending it by continuing the perturbation process until reaching a specified maximum perturbation magnitude. The objective of the PGD attack is to find the smallest perturbation that leads to misclassification while constraining the perturbed example to remain within a specified L_p-norm distance from the original input. The PGD attack starts with the original input image and iteratively computes the perturbation required to induce misclassification. At each iteration, it calculates the gradient of the loss function with respect to the input and applies a small step in the direction that maximizes the loss while ensuring the perturbed example remains within the specified L_p-norm ball around the original input. This process continues for a predetermined number of iterations or until a misclassification is achieved. Mathematically, the perturbed input at each iteration of the PGD attack is computed as follows:

$$x_{\text{adv}}^{s} = \text{clip}\Big(x_{\text{adv}}^{s-1} + \alpha\,\text{sign}\big(\nabla_x J(w^\top x_{\text{clean}}, y)\big), \\ x_{\text{adv}} - \epsilon,\, x_{\text{adv}} + \epsilon\Big)$$

where x_{adv}^{t-1} is the input image at the current iteration t, α is the step size, $\nabla_{\text{x}} J(w^T x_{\text{clean}}, y)$ is the gradient of the loss function with respect to the input image x_{clean}, and clip function ensures that the perturbed image remains within a specified range defined by the lower and upper bounds.

- **Carlini-Wagner** (C&W, L_2, Untargeted): The Carlini-Wagner (C&W) attack [22], introduced by Carlini and Wagner in 2017, is a powerful optimization-based method for crafting adversarial examples. Unlike many other attack methods that focus on adding imperceptible perturbations to input data, the C&W attack formulates the attack as an optimization problem aimed at finding the smallest perturbation that leads to misclassification while satisfying certain constraints. The objective of the C&W attack is to find a perturbation δ that minimizes a combination of the perturbation magnitude and a loss function, subject to various constraints. The loss function is typically designed to encourage misclassification while penalizing large perturbations. The constraints ensure that the perturbed example remains within a specified L_p-norm distance from the original input and maintains perceptual similarity. The objective function of the C&W attack can be formulated as follows:

$$\min \|\delta\|_l + c \cdot f(x_{\text{clean}} + \delta)$$

where $\|\delta\|_l$ represents the L_l-norm of the perturbation, $f(x_{\text{clean}} + \delta)$ is the loss function representing misclassification, and c is a regularization parameter that balances the trade-off between the perturbation magnitude and the loss function.

B. Analysis

In this section, we provide an explanation for why our proposed `MirrorCheck` pipeline is robust to both *non-adaptive* and *adaptive* adversarial attacks. We begin by propose `MirrorCheck` as an autoencoder before defining the overall setting, threat model, and key assumptions. We then present our main theorem and proof sketch, demonstrating why clean images remain undetected while adversarial ones are flagged with high probability. Finally, we extend the argument to the *adaptive* setting, where the attacker has knowledge of our detection pipeline and attempts to circumvent it by making the entire process differentiable.

B.1. `MirrorCheck` as an Autoencoder

In the auto-encoder literature, reconstruction error has been shown to be a reliable indicator of whether a sample is in or out of the training distribution [78–80]. We now cast `MirrorCheck` as a particular kind of auto-encoder to leverage these results and justify our approach. `MirrorCheck` can be conceptualized within the structure of regular [81–83] and Variational Autoencoders (VAEs) [84–86], which typically encode input data into a continuous latent space through an encoder and reconstruct the input using a decoder. Unlike typical variational-autoencoders, `MirrorCheck` relies on a discrete, categorical latent space comprising textual descriptions generated from images. In this respect, it is in line with recent VAEs that incorporate categorical latent variables through mechanisms such as the Gumbel-Softmax distribution [87–91].

The I2T phase of `MirrorCheck` acts as the encoder, mapping high-dimensional visual data into a discrete latent space represented by text. This process can be mathematically expressed as

$$q_\phi(\mathbf{z}|\mathbf{x}) = \text{Cat}(\mathbf{z}; \boldsymbol{\pi}(\mathbf{x}))\ , \tag{14}$$

where $\mathbf{x}$ is the input image, $\mathbf{z}$ represents the latent textual description, Cat denotes the categorical distribution, and $\boldsymbol{\pi}(\mathbf{x})$ is the distribution over the discrete latent variables conditioned on the input image, parameterized by ϕ.
The T2I phase serves as the decoder. It reconstructs the visual data from these textual descriptions. It can be written as

$$p_\theta(\mathbf{x}|\mathbf{z}) = \text{Bernoulli}(\mathbf{x}; \boldsymbol{\sigma}(\mathbf{z})), \tag{15}$$

where $\boldsymbol{\sigma}(\mathbf{z})$ models the probability of generating an image $\mathbf{x}$ from the latent description $\mathbf{z}$, parameterized by θ. When sampling caption text with a non-zero softmax temperature,

these steps resemble the Gumbel-Softmax reparameterization trick, typically used in Variational Autoencoders (VAEs) to sample from the latent [87, 88].

Thus, using the reconstruction error as an indication of whether an input has been compromised via an adversarial attack is as justified as using it to determine if a sample is out-of-distribution when employing a VAE. This aligns with earlier work [30, 59] that showed that this metric is good at detecting adversarial attacks. It is also in the same spirit as approaches to detecting anomalies through segmentation and reconstruction [92, 93].

B.2. Key Observations

1. *Clean Consistency.* A small perturbation from x_{clean} typically does not change the VLM's caption drastically. If the caption remains accurate, then G_ψ produces a corresponding $x_{\text{gen,clean}}$ that is semantically aligned with x_{clean}. Finally, by Lipschitzness, these two images remain close in the embedding space, pushing $\text{sim}(\cdot,\cdot)$ above τ^*.
2. *Adversarial Detection.* If the adversary's perturbation δ drastically changes the VLM's output caption t_{adv}, then $x_{\text{gen,adv}}$ becomes semantically inconsistent with x_{adv}. In embedding space, $\text{sim}\big(\mathcal{I}_{\hat{\phi}}(x_{\text{adv}}), \mathcal{I}_{\hat{\phi}}(x_{\text{gen,adv}})\big)$ plummets below τ^*, triggering detection.
3. *Randomization / Ensemble.* Even if the adversary tries to *adapt* by directly optimizing similarity, randomization or multiple encoders ensure that the gradient alignment is broken. The attacker cannot perfectly maintain high similarity under *all* encoders, especially if one-time noise (OTU) is added right before inference.
4. *Ensemble Randomization Fractures Gradient Alignment.* Different encoders $\{\mathcal{I}_{\phi j}\}$ exhibit distinct embedding geometries; a single perturbation δ usually cannot keep all similarity scores high at once. Unless $\|\delta\|$ is made tiny (reducing the adversarial effect), mismatch arises in at least one encoder, causing detection.
5. *OTU Noise Breaks Perfect Differentiability.* If the defender applies random perturbations γ to ϕ right before detection, the attacker's precomputed gradient w.r.t. ϕ no longer matches the final inference pass. Thus, any carefully crafted δ may fail to preserve similarity under the *real* detector.
6. *Semantic Mismatch Argument Persists.* Even with a continuous pipeline, forcing a drastically different caption (to meet the adversarial goal) yields a T2I-generated image that is semantically far from x_{adv}. The attacker faces a contradiction between requiring *large semantic drift* (to produce a malicious caption) and *small semantic drift* (to retain high similarity). They typically cannot satisfy both simultaneously.

Table 4. Similarity scores. The average shows that `MirrorCheck` is able to maximize the difference between clean and adversarial images for all victim models.

Victim Model	Setting	RN50			RN101			ViT-B/16			ViT-B/32			ViT-L/14		
		Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
UniDiffuser	Clean	**0.720**	0.241	0.931	**0.818**	0.512	0.963	**0.758**	0.320	0.975	0.750	0.344	0.973	**0.723**	0.244	0.952
	AttackVLM-T	0.414	0.118	0.872	0.628	0.434	0.938	0.515	0.222	0.852	**0.807**	0.426	0.958	0.516	0.130	0.820
	AttackVLM-Q	0.421	0.165	0.742	0.676	0.539	0.780	0.551	0.330	0.759	0.528	0.274	0.725	0.547	0.280	0.735
BLIP	Clean	**0.699**	0.162	0.911	**0.804**	0.434	0.953	**0.741**	0.247	0.948	**0.723**	0.222	0.945	**0.705**	0.126	0.944
	AttackVLM-T	0.395	0.077	0.823	0.627	0.455	0.858	0.522	0.239	0.847	0.487	0.173	0.798	0.512	0.070	0.828
	AttackVLM-Q	0.443	0.165	0.694	0.679	0.522	0.81	0.563	0.276	0.740	0.534	0.212	0.750	0.561	0.277	0.757
BLIP-2	Clean	**0.712**	0.151	0.936	**0.813**	0.422	0.965	**0.757**	0.248	0.961	**0.737**	0.213	0.946	**0.725**	0.189	0.948
	AttackVLM-T	0.439	0.045	0.827	0.644	0.417	0.884	0.543	0.218	0.864	0.498	0.175	0.844	0.544	0.140	0.822
	AttackVLM-Q	0.409	0.124	0.684	0.668	0.488	0.791	0.538	0.316	0.746	0.519	0.301	0.721	0.530	0.249	0.734
Img2Prompt	Clean	**0.652**	0.212	0.912	**0.775**	0.454	0.946	**0.699**	0.297	0.949	**0.684**	0.236	0.93	**0.667**	0.151	0.939
	AttackVLM-T	0.389	0.097	0.798	0.626	0.426	0.866	0.517	0.214	0.822	0.481	0.161	0.797	0.508	0.129	0.794
	AttackVLM-Q	0.448	0.116	0.698	0.683	0.501	0.820	0.564	0.316	0.731	0.536	0.240	0.761	0.563	0.270	0.801

Table 5. Detection accuracies. TPR is the proportion of actual adversarial images that are correctly identified. FPR is the proportion of clean images incorrectly identified as adversarial. Accuracy is the proportion of correctly identified images (both clean and adversarial).

Victim Model	Setting	RN50			RN101			ViT-B/16			ViT-B/32			ViT-L/14			Ensemble		
		TPR	FPR	ACC	TPR	FPR	ACC	TPR	FPR	ACC	TPR	FPR	ACC	TPR	FPR	ACC	TPR	FPR	ACC
UniDiffuser	AttackVLM-T	0.917	0.085	**0.916**	0.912	0.088	0.912	0.902	0.098	0.902	0.368	0.636	0.366	0.874	0.127	0.874	0.87	0.13	0.87
	AttackVLM-Q	**0.925**	0.075	0.925	0.871	0.129	0.871	0.874	0.125	0.875	0.889	0.113	0.888	0.825	0.174	0.826	0.895	0.105	0.895
BLIP	AttackVLM-T	0.905	0.096	**0.905**	0.894	0.108	0.893	0.876	0.126	0.875	0.887	0.114	0.887	0.84	0.159	0.841	0.898	0.103	0.898
	AttackVLM-Q	0.896	0.104	**0.896**	0.855	0.144	0.856	0.838	0.162	0.838	0.854	0.144	0.855	0.792	0.213	0.790	0.865	0.136	0.865
BLIP-2	AttackVLM-T	0.882	0.119	0.882	0.883	0.117	0.883	0.873	0.128	0.873	0.898	0.102	**0.898**	0.835	0.166	0.835	0.891	0.111	0.890
	AttackVLM-Q	0.921	0.082	**0.920**	0.885	0.117	0.884	0.886	0.114	0.886	0.896	0.104	0.896	0.856	0.144	0.856	0.912	0.090	0.911
Img2Prompt	AttackVLM-T	0.841	0.160	**0.841**	0.833	0.170	0.832	0.815	0.185	0.815	0.838	0.164	0.837	0.783	0.216	0.784	0.834	0.167	0.8335
	AttackVLM-Q	0.809	0.195	**0.807**	0.759	0.242	0.7585	0.767	0.235	0.766	0.789	0.213	0.788	0.708	0.295	0.707	0.782	0.220	0.781

Table 6. Similarity Scores for `MirrorCheck` using UniDiffuser as our T2I model and CLIP as the encoder.

Victim Model	Setting	CLIP Image Encoder					
		RN50	RN101	ViT-B/16	ViT-B/32	ViT-L/14	Ensemble
UniDiffuser [39]	Clean	**0.737**	**0.826**	**0.769**	0.764	**0.721**	**0.763**
	AttackVLM-T	0.408	0.617	0.501	**0.765**	0.486	0.555
	AttackVLM-Q	0.396	0.659	0.526	0.508	0.520	0.522
BLIP [2]	Clean	**0.713**	**0.806**	**0.742**	**0.730**	**0.685**	**0.735**
	AttackVLM-T	0.375	0.609	0.500	0.466	0.480	0.486
	AttackVLM-Q	0.417	0.656	0.529	0.503	0.526	0.526
BLIP-2 [3]	Clean	**0.732**	**0.823**	**0.764**	**0.759**	**0.720**	**0.760**
	AttackVLM-T	0.425	0.627	0.533	0.491	0.517	0.519
	AttackVLM-Q	0.390	0.652	0.511	0.506	0.510	0.514
Img2Prompt [40]	Clean	**0.663**	**0.780**	**0.703**	**0.689**	**0.660**	**0.699**
	AttackVLM-T	0.369	0.607	0.494	0.457	0.474	0.480
	AttackVLM-Q	0.417	0.656	0.522	0.502	0.525	0.525
MiniGPT-4 [4]	Clean	**0.599**	**0.737**	**0.646**	**0.641**	**0.610**	**0.646**
	AttackVLM-T	0.507	0.678	0.570	0.540	0.524	0.564

C. Additional Empirical Results

Generalization across Encoders and T2I Models. We present the results of the vanilla variant of `MirrorCheck`. Tab. 4 and Tab. 5 shows the performance. Here, we leverage UniDiffuser T2I model [1] and ControlNet [54]. We observe better accuracies using UniDiffuser, compared to using Stable Diffusion. We also observe better accuracies using ControlNet, compared to using Stable Diffusion, and slightly better overall accuracies compared to UniDiffuser. Tab. 6 and Tab. 8 show the similarities when using UniDiffuser-T2I [1] and ControlNet [54] for image generation and the CLIP models for evaluation, while Tab. 7 and Tab. 9 show the detection accuracies. Overall, we show that our approach is agnostic of and generalizes across encoders and T2I models.

Table 7. Detection performance for MirrorCheck using UniDiffuser as our T2I model and CLIP as the encoder.

Victim Model	Setting	CLIP Image Encoders					
		RN50	RN101	ViT-B/16	ViT-B/32	ViT-L/14	Ensemble
UniDiffuser [39]	AttackVLM-T	**0.935**	0.910	0.910	0.470	0.910	0.827
	AttackVLM-Q	**0.960**	0.905	0.900	0.920	0.865	0.909
BLIP [2]	AttackVLM-T	0.915	0.910	0.915	**0.920**	0.845	0.901
	AttackVLM-Q	**0.920**	0.880	0.900	0.915	0.820	0.887
BLIP-2 [3]	AttackVLM-T	0.915	0.930	0.885	**0.935**	0.860	0.905
	AttackVLM-Q	**0.950**	0.910	0.920	0.930	0.860	0.914
Img2Prompt [40]	AttackVLM-T	**0.885**	0.870	0.830	**0.885**	0.810	0.856
	AttackVLM-Q	**0.845**	0.810	0.805	0.830	0.775	0.813

Table 8. Similarity Scores for MirrorCheck using ControlNet as our T2I model and CLIP as the encoder.

Victim Model	Setting	CLIP Image Encoder					
		RN50	RN101	ViT-B/16	ViT-B/32	ViT-L/14	Ensemble
UniDiffuser [39]	Clean Image	**0.747**	**0.839**	**0.768**	**0.758**	**0.731**	**0.769**
	AttackVLM-T	0.410	0.621	0.514	0.554	0.514	0.523
	AttackVLM-Q	0.440	0.663	0.555	0.522	0.519	0.540
BLIP [2]	Clean Image	**0.747**	**0.840**	**0.770**	**0.769**	**0.728**	**0.770**
	AttackVLM-T	0.508	0.625	0.526	0.484	0.511	0.511
	AttackVLM-Q	0.466	0.689	0.575	0.527	0.565	0.564
BLIP-2 [3]	Clean Image	**0.751**	**0.844**	**0.774**	**0.766**	**0.735**	**0.774**
	AttackVLM-T	0.388	0.623	0.526	0.493	0.512	0.508
	AttackVLM-Q	0.463	0.684	0.571	0.522	0.565	0.561
Img2Prompt [40]	Clean Image	**0.661**	**0.780**	**0.712**	**0.695**	**0.670**	**0.703**
	AttackVLM-T	0.400	0.626	0.532	0.497	0.514	0.514
	AttackVLM-Q	0.463	0.685	0.569	0.534	0.569	0.564

Table 9. Detection Performance for MirrorCheck using ControlNet as our T2I model and CLIP as the encoder.

Victim Model	Setting	CLIP Image Encoder					
		RN50	RN101	ViT-B/16	ViT-B/32	ViT-L/14	Ensemble
UniDiffuser [39]	AttackVLM-T	0.935	**0.980**	0.925	0.895	0.920	0.931
	AttackVLM-Q	0.945	**0.965**	0.880	0.920	0.880	0.918
BLIP [2]	AttackVLM-T	0.955	**0.965**	0.880	0.945	0.880	0.925
	AttackVLM-Q	**0.940**	0.905	0.870	0.925	0.830	0.894
BLIP-2 [3]	AttackVLM-T	0.935	**0.950**	0.905	0.930	0.900	0.924
	AttackVLM-Q	**0.915**	0.910	0.880	0.890	0.850	0.889
Img2Prompt [40]	AttackVLM-T	**0.965**	0.940	0.900	0.950	0.890	0.929
	AttackVLM-Q	**0.950**	0.895	0.860	0.915	0.800	0.884

Robustness to Adaptive Attacks. To further evaluate the robustness of MirrorCheck against adaptive adversaries, we introduce an additional adaptive attack similar to the BPDA+EOT attack discussed in the main paper, but with a different optimization objective. This attack constructs a fully differentiable end-to-end pipeline by linking the victim vision-language model (VLM) and a text-to-image (T2I) generative model through a learned adapter network. The attacker's objective is to generate an adversarial image x_{adv} that simultaneously aligns with the target caption t^* and its corresponding generated image x_{gen}, thereby attempting to bypass our detection mechanism. The optimization objective (Algorithm 1) enforces similarity between these representations while employing randomized encoders and Expectation over Transformation (EOT) to approximate gradients through the defense. We assess the attacker's success under varying levels of knowledge about the image encoders used in MirrorCheck.

As shown in Tab. 10, detection performance improves as the number of encoders increases and when stochasticity (OTU approach) is applied, significantly hindering the attacker's ability to evade detection. Even under the strongest assumption, where the attacker knows all encoders, detection accuracy remains high. Moreover, we also compared text-embedding similarities between target and generated captions for standard (ADV-Transfer) and adaptive attacks Tab. 11. The adaptive attacks yield notably lower similarity scores, demonstrating reduced attack success. These results confirm that MirrorCheck maintains strong robustness even against highly adaptive, gradient-based attacks, benefiting from encoder diversity and stochastic defense components.

Algorithm 1 Adaptive Attack using Learnable Adapters

Input: Original image x_{in}, target caption t
Output: Adversarial image x_{adv}
Initialize: $\delta \leftarrow 0$
VLM Model: $\mathcal{F}_\theta(x_{\text{in}}; p) \rightarrow t$ ▷ Victim model generates caption
VLM Text Encoder: $\hat{\mathcal{F}}_\theta(x_{\text{in}}) \rightarrow z$
T2I Image Generator: $\hat{G}_\psi(z) \rightarrow x_{gen}$
Adapter Network Training: Train adapter $\mathcal{A}$
repeat
  $x_{adv} \leftarrow x_{in} + \delta$
  $z \leftarrow \hat{\mathcal{F}}_\theta(x_{adv})$
  $z' \leftarrow \mathcal{A}(z)$
  $x_{gen} \leftarrow \hat{G}_\psi(z')$
  for $j = 1$ to N **do**
    $v_{\text{adv}}^{(j)} \leftarrow \mathcal{I}_{\phi j,\xi}(x_{adv})$
    $v_{\text{gen}}^{(j)} \leftarrow \mathcal{I}_{\phi j,\xi}(x_{gen})$
  end for
  Compute broken-down loss terms:
    $L_{\text{img-target}} \leftarrow d\big(\tilde{\mathcal{I}}_\phi(x_{adv}), \tilde{\mathcal{I}}_\phi(G_\psi(t^*; \eta))\big)$
    $L_{\text{adv-gen}} \leftarrow \frac{1}{N}\sum_{j=1}^{N} d\big(v_{\text{adv}}^{(j)}, v_{\text{gen}}^{(j)}\big)$
    $L_{\text{total}} \leftarrow L_{\text{img-target}} + L_{\text{adv-gen}}$
  Update δ**:**
    $\delta \leftarrow \delta - \gamma \cdot \nabla_\delta L_{\text{total}}$
until Convergence
$x_{\text{adv}} \leftarrow x_{in} + \delta$
return x_{adv}

Table 10. Robustness of MirrorCheck on adversarial samples generated through adaptive attacks based on the attacker's knowledge of image encoders used in MirrorCheck. The defender employs between one and five pretrained CLIP image encoders with backbones RN50, RN101, ViT-B/16, ViT-B/32, and ViT-L/14. The attacker has knowledge of all, all but one, or all but two of these encoders, and replaces unknown ones with OpenCLIP encoders.

Attacked Image Encoder	MirrorCheck			MirrorCheck (OTU approach)		
	ALL	ALL but ONE	ALL but TWO	ALL	ALL but ONE	ALL but TWO
ViT-B/32	0.55	0.90	-	0.50	0.90	-
RN50 and ViT-B/32	0.60	0.70	0.90	0.70	0.80	0.90
RN50, ViT-B/32, and ViT-L/14	0.65	0.65	0.80	0.75	0.75	0.80
RN50, ViT-B/16, ViT-B/32, and ViT-L/14	0.65	0.65	0.85	0.75	0.80	0.85
RN50, RN101, ViT-B/16, ViT-B/32, and ViT-L/14	0.75	0.75	0.85	0.85	0.90	0.80

Impact of Clean Ratio on Detection Accuracy. Fig. 4 illustrates how varying the proportion of clean to adversar-

Table 11. Text embedding similarity between the target captions and captions produced by the victim model under transfer (ADV-Transfer) and adaptive attacks. Lower similarity indicates stronger robustness.

Victim Model	Setting	CLIP Image Encoder					
		RN50	RN101	ViT-B/16	ViT-B/32	ViT-L/14	Ensemble
UniDiffuser	ADV-Transfer	0.76	0.71	0.74	0.77	0.68	0.73
	Adaptive (ViT-B/32)	0.59	0.61	0.60	0.64	0.53	0.60
	Adaptive (RN50 + ViT-B/32)	0.63	0.60	0.63	0.66	0.55	0.61
	Adaptive (RN50 + ViT-B/32 + ViT-L/14)	0.70	0.64	0.68	0.70	0.60	0.66
	Adaptive (RN50 + ViT-B/16 + ViT-B/32 + ViT-L/14)	0.69	0.64	0.68	0.71	0.62	0.67
	Adaptive (RN50 + RN101 + ViT-B/16 + ViT-B/32 + ViT-L/14)	0.63	0.64	0.66	0.67	0.58	0.64

ial examples affects detection accuracy. As the clean ratio increases from 50% to 99.9%, overall performance consistently improves. This trend is most evident for the RN50 encoder, which maintains strong ROC AUC scores even at lower clean ratios. In contrast, encoders such as ViT-L/14 are more sensitive to reduced clean ratios, exhibiting a noticeable performance drop, particularly near the 99% level. These observations indicate that some encoders are inherently more robust to imbalanced datasets. Our ensemble approach effectively mitigates these disparities, combining the strengths of different encoders to deliver stable, well-rounded performance. Notably, detection performance stabilizes at the highest clean ratio (99.9%), where all encoders achieve their best or near-best results. Overall, these findings demonstrate that our method remains reliable and accurate across a wide range of clean-to-adversarial distributions, even when adversarial interference is minimal.

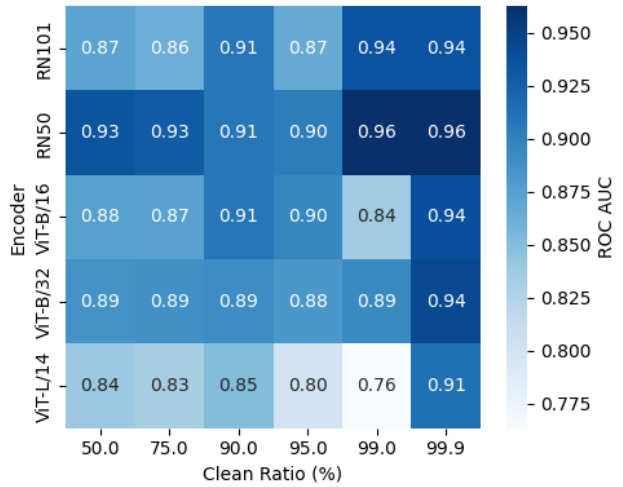


Figure 4. Effect of Clean Ratio on Detection Accuracy across Different Encoders.

D. Qualitative Examples

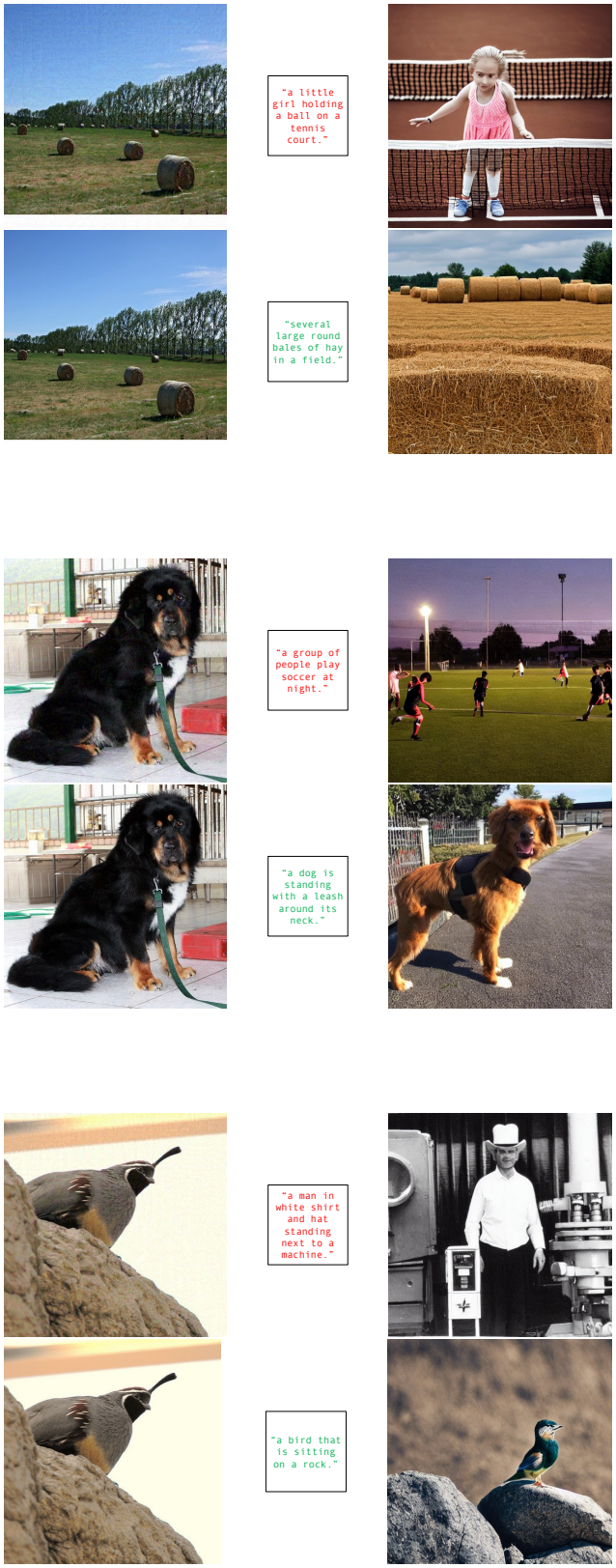


Figure 5. Visual results using BLIP (Victim Model) and Stable Diffusion (T2I Model).

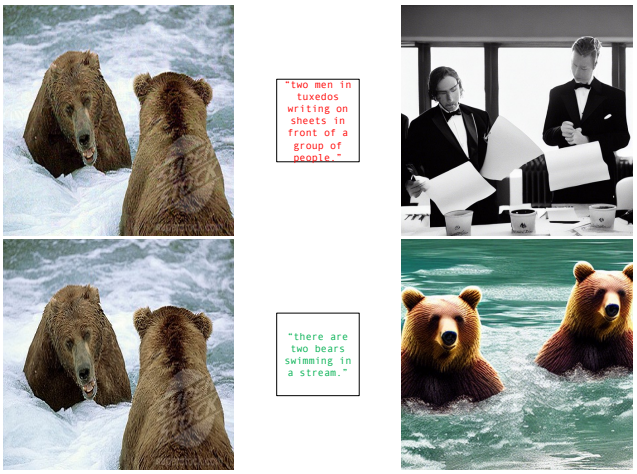
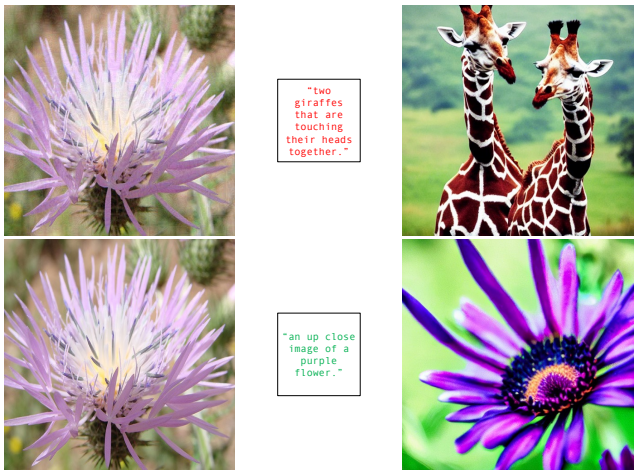
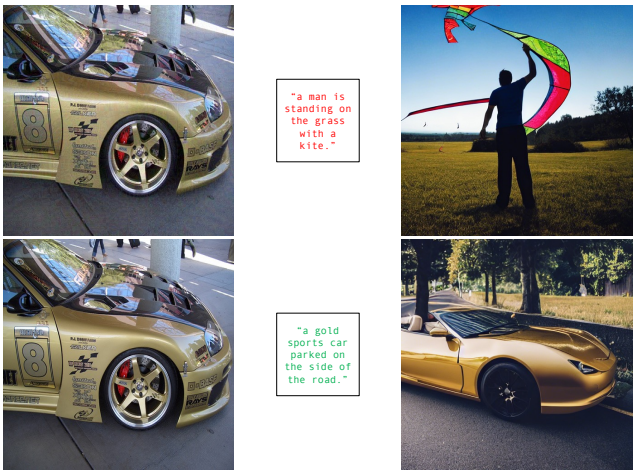
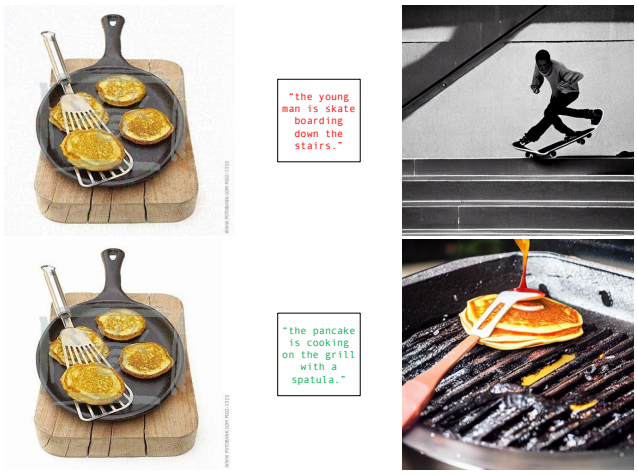


Figure 6. Visual results using BLIP (Victim Model) and Stable Diffusion (T2I Model).







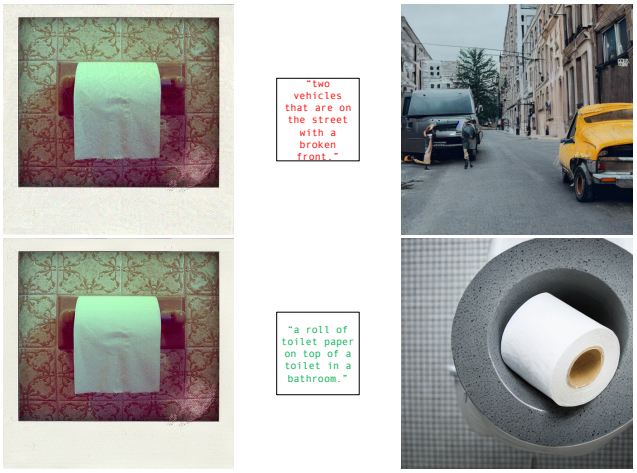


Figure 7. Visual results using BLIP (Victim Model) and Stable Diffusion (T2I Model).