

# AI-generated Image Detection: Passive or Watermark?

Moyang Guo<sup>\*,1</sup>, Yuepeng Hu<sup>\*,1</sup>, Zhengyuan Jiang<sup>\*,1</sup>, Zeyu Li<sup>\*,1</sup>, Amir Sadovnik<sup>2</sup>, Arka Daw<sup>2</sup>, Neil Gong<sup>1</sup>

<sup>1</sup>Duke University

{moyang.guo, yuepeng.hu, zhengyuan.jiang, zeyu.li030, neil.gong}@duke.edu

<sup>2</sup>Oak Ridge National Laboratory

{sadovnika, dawa}@ornl.gov

## ABSTRACT

While text-to-image models offer numerous benefits, they also pose significant societal risks. Detecting AI-generated images is crucial for mitigating these risks. Detection methods can be broadly categorized into *passive* and *watermark-based* approaches: passive detectors rely on artifacts present in AI-generated images, whereas watermark-based detectors proactively embed watermarks into such images. A key question is which type of detector performs better in terms of effectiveness, robustness, and efficiency. However, the current literature lacks a comprehensive understanding of this issue. In this work, we aim to bridge that gap by developing *ImageDetectBench*, the first comprehensive benchmark to compare the effectiveness, robustness, and efficiency of passive and watermark-based detectors. Our benchmark includes four datasets, each containing a mix of AI-generated and non-AI-generated images. We evaluate five passive detectors and four watermark-based detectors against eight types of common perturbations and three types of adversarial perturbations. Our benchmark results reveal several interesting findings. For instance, watermark-based detectors consistently outperform passive detectors, both in the presence and absence of perturbations. Based on these insights, we provide recommendations for detecting AI-generated images, e.g., when both types of detectors are applicable, watermark-based detectors should be the preferred choice.

## 1 Introduction

Text-to-image models [1–4] can generate highly realistic images from simple text descriptions called *prompts*, offering substantial benefits across multiple domains. In content creation, they enable individuals with limited graphic design skills or resources to produce high-quality images; in education, they provide customized imagery to aid learning and engagement; and in marketing and design, they empower creators to generate on-demand, tailored visuals, reducing time and cost barriers. As a result, these capabilities have attracted significant interest from both academic researchers and industry developers.

However, alongside these benefits, text-to-image models also introduce severe societal risks. By enabling anyone to create realistic images, these models can be misused to spread misinformation and cause public harms. For instance, AI-generated images of prominent figures, like those depicting fictitious events involving public figures such as Joe Biden [5] or Donald Trump [6], have been used to manipulate public opinion or disseminate false information. Such instances raise ethical and regulatory concerns about the responsible use of these models. Addressing these risks is crucial, especially as AI-generated images become more indistinguishable from real ones.

Detecting AI-generated images has become a crucial technology for mitigating these societal risks. At its core, a detector functions as a binary classifier, taking an image as input and determining whether it is AI-generated or non-AI-generated. Based on their design, detectors can be categorized into two types: *passive* and *watermark-based*. Passive detectors [7–14] rely on identifying subtle artifacts inherent to AI-generated images that may differ from non-AI-generated ones. They leverage standard deep learning classifiers to classify images based on features extracted from their pixel values. For instance, companies like [hive.ai](#) [13] and [illuminarty.ai](#) [14] offer passive detection services to identify AI-generated images. By contrast, watermark-based detectors [15–22] proactively embed a human-

---

\*These authors contributed equally and are ordered alphabetically by last name.

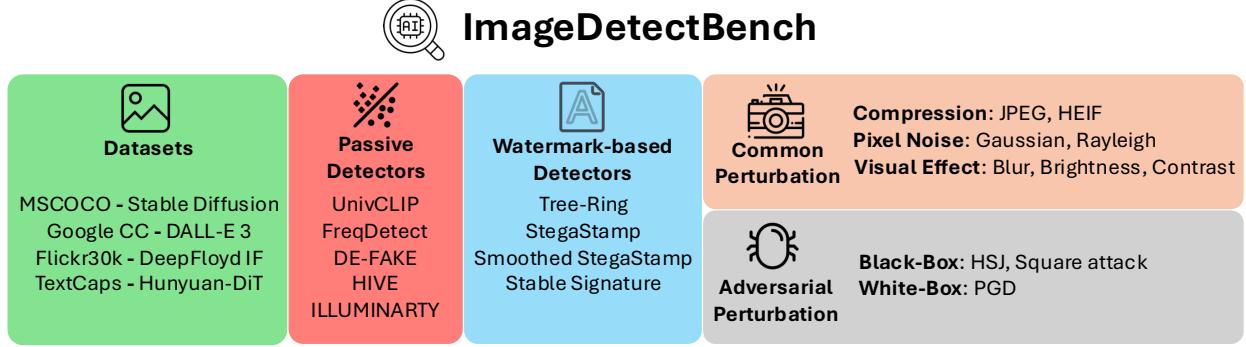


Figure 1: Overview of our ImageDetectBench.

imperceptible “artifact” (called watermark) into images during their generation. An image is detected as AI-generated if a similar watermark can be extracted from it. Watermark has recently gained substantial attention in both policy and industry; for example, the Executive Order on Trustworthy Artificial Intelligence from the White House [23] recommends watermarking AI-generated content. Several major technology companies—such as OpenAI [24], Google [25], Meta [18], and Microsoft [26]—have also deployed watermarks.

While both detectors are actively deployed, there remains a key question: Which type of detector is more effective, robust, and efficient under real-world conditions? Understanding the comparative strengths and weaknesses of these two types of detectors is crucial for advancing detection technology and reducing the potential harms associated with AI-generated images. However, existing research primarily focuses on evaluating the effectiveness and robustness of either passive [27–30] or watermark-based detectors [16, 22, 31–34], leaving a comparative analysis under the same threat models largely unexplored.

**Our work:** We aim to bridge this gap by proposing ImageDetectBench, the *first* comprehensive benchmark for the comparative evaluation of passive and watermark-based detectors across diverse datasets and image perturbations. Figure 1 provides an overview of our ImageDetectBench.

- **Datasets.** ImageDetectBench includes four diverse datasets, each with distinct image characteristics. Every dataset contains an equal number of non-AI-generated and AI-generated images. We collected the non-AI-generated images from various established image benchmark datasets, while creating the AI-generated images using four text-to-image models. To reduce the influence of image content on the distinction between AI-generated and non-AI-generated images, we used the captions of the non-AI-generated images as prompts to create the AI-generated images.

- **Perturbations.** An attacker may deceive a detector through either a *removal* or *forgery* attack. In a removal attack, the goal is to make the detector misclassify an AI-generated image as non-AI-generated, while in a forgery attack, a non-AI-generated image is misclassified as AI-generated. To assess the robustness of both passive and watermark-based detectors against these threats, ImageDetectBench includes a variety of image perturbations. These consist of eight *common perturbations* [35–39] and three *adversarial perturbations* (two black-box [40, 41] and one white-box [42]). Common perturbations encompass widely used image processing operations, such as JPEG compression, which may be applied by regular users as well as attackers. In contrast, adversarial perturbations are intentionally crafted to deceive detectors.

- **Systematic benchmarking, findings, and recommendations.** Using our datasets and perturbations, we systematically benchmark the comparative effectiveness, robustness, and efficiency of five state-of-the-art passive detectors [8–10, 13, 14] and four watermark-based detectors [15–18] under consistent threat models. Our comprehensive evaluation reveals several key findings. Notably, watermark-based detectors consistently outperform passive detectors across all scenarios, including those with no perturbations, common perturbations, and adversarial perturbations. Moreover, watermark-based detectors demonstrate remarkable insensitivity to variations in image distributions, whereas passive detectors are significantly impacted. Lastly, the most robust watermark-based detector is two orders of magnitude faster than the most robust passive detector. Based on these findings, we offer several recommendations. For instance, when both detector types are applicable, watermark-based detectors should be prioritized for their superior effectiveness, robustness, and efficiency.

To summarize, our key contributions are as follows:

- We propose ImageDetectBench, the *first* benchmark designed to systematically compare the effectiveness, robustness, and efficiency of passive and watermark-based AI-generated image detectors.

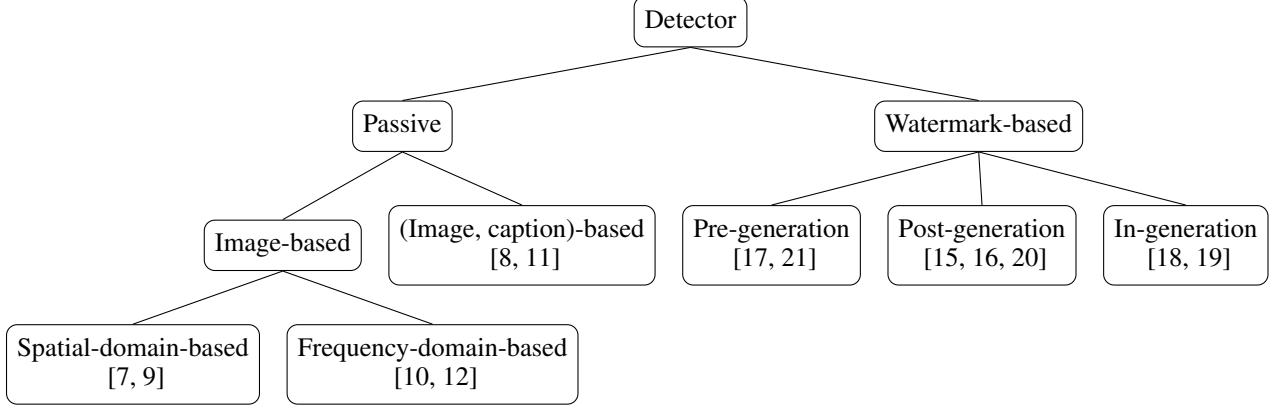


Figure 2: Our taxonomy of detectors for AI-generated image detection.

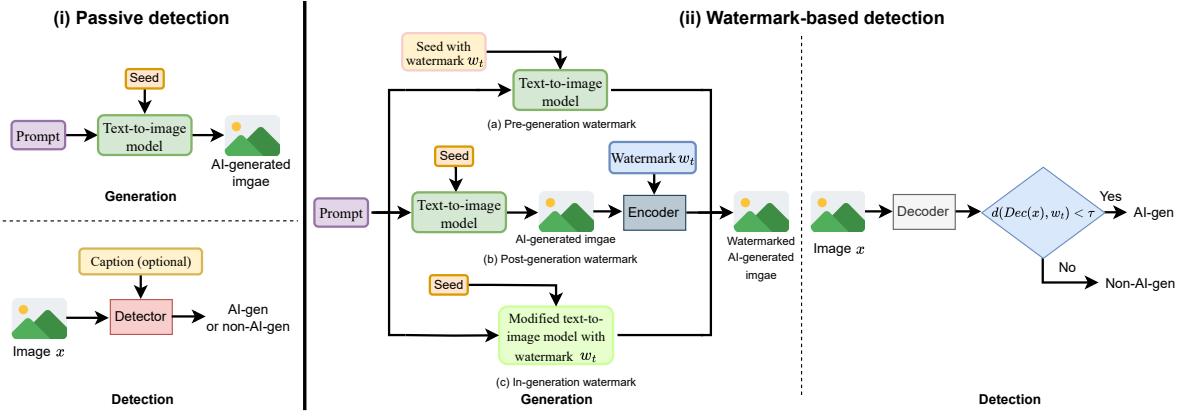


Figure 3: Illustration of passive and watermark-based detection of AI-generated images.

- ImageDetectBench incorporates four diverse datasets, eleven types of perturbations (eight common perturbations and three adversarial perturbations), and a total of nine detectors: five passive and four watermark-based.
- Based on our comprehensive benchmark study, we present key findings and offer recommendations for AI-generated image detection.

## 2 Taxonomy of Detectors

A detector aims to classify images as either AI-generated or non-AI-generated. As illustrated in Figure 2, detection methods can be divided into two categories: *passive* [7–14] and *watermark-based* [15–21]. Passive detectors identify AI-generated images by analyzing artifacts present within the images, whereas watermark-based detectors proactively embed a watermark into an AI-generated image, enabling later detection. Figure 3 illustrates the difference between passive and watermark-based detection. We denote a detector as  $D$ , which takes an image  $x$  as input and outputs either 1 or 0, where  $D(x) = 1$  indicates  $x$  is AI-generated and  $D(x) = 0$  indicates  $x$  is non-AI-generated. We note that some passive detectors [8, 11] also incorporate an associated caption of the image  $x$ . In this work, we mainly focus on detectors from academic research since they are public but we also evaluate commercial detectors [13, 14] in Section 6.6.

### 2.1 Passive Detectors

Passive detection methods train binary classifiers by following a standard machine learning pipeline. The initial step is to collect a training dataset comprising both AI-generated and non-AI-generated images. Some passive detection methods [8, 11] also require a caption for each image in the training dataset. During training, features of images (and captions, if applicable) are extracted and used to train a binary classifier through supervised learning. During testing,

given an image (and optionally its caption), features are extracted and fed into the binary classifier to identify whether this image is AI-generated or not. Different passive detectors essentially use different features and/or binary classifiers. Based on whether captions are utilized as input, passive detectors can be further classified into two subcategories: *image-based* and *(image, caption)-based*.

**Image-based detectors:** Image-based detectors [7, 9, 10, 12] rely solely on images as input and are typically categorized as *spatial-domain-based* or *frequency-domain-based*. Spatial-domain-based detectors [7, 9] extract features in the spatial domain for detection. In contrast, frequency-domain-based detectors [10, 12] transform an image into the frequency domain, leveraging frequency-domain features to identify AI-generated images. In the following, we pick one representative detector from each category to discuss more details.

- **UnivCLIP [9].** UnivCLIP is a spatial-domain-based detector that leverages high-resolution pretrained image encoders, such as CLIP’s Vision Transformer (ViT), to extract spatial features from images. The features extracted by the CLIP encoder are passed through a single linear layer with sigmoid activation for classification.
- **Frequency Domain Detection (FreqDetect) [10].** FreqDetect is a frequency-domain-based detector. This detector extracts frequency features using transforms such as the Discrete Cosine Transform (DCT) and classifies them with logistic regression. Specifically, FreqDetect first converts an image to grayscale. The DCT then transforms the grayscale pixel values into a matrix of frequency coefficients. This matrix is flattened to serve as the feature vector for classification.

**(Image, caption)-based detectors:** These detectors [8, 11] take both an image and a caption as input, leveraging the combined information from textual and visual data. The key assumption is that AI-generated images are likely to closely align with the content described in their associated prompts/captions, whereas non-AI-generated images often contain additional details that may not be explicitly described by a caption. In our experiments, we use DE-FAKE [8] as an example detector in this subcategory.

- **DE-FAKE [8].** DE-FAKE uses CLIP’s image and text encoders to extract features from an image and its caption, respectively. By concatenating these features, DE-FAKE creates a unified feature vector representing each image-caption pair. The classifier in DE-FAKE is a three-layer multilayer perceptron. During training, a set of (non-AI-generated image, caption) pairs from standard benchmark datasets and (AI-generated image, prompt) pairs are used to train the classifier, where the prompt used to generate an AI image is treated as its caption. During testing, if a caption is unavailable for a given image, it is generated by a captioning model, such as BLIP [43].

## 2.2 Watermark-based Detectors

In contrast to passive detection, watermark-based detection proactively embeds watermarks in AI-generated images to facilitate their detection. In watermark-based detection, all AI-generated images are watermarked, while non-AI-generated images remain non-watermarked. The core of image watermarking methods lies in the *embedding* and *detection* of these watermarks. Based on the stage at which the watermark is embedded into an AI-generated image, image watermarks are categorized into three types: *pre-generation* [17, 21], *post-generation* [15, 16, 20], and *in-generation* [18, 19]. Pre-generation methods modify the seed of the image generation process so that the image generated with the specific seed has a watermark. Post-generation methods add watermarks after generation by subtly modifying pixel values in the image. In-generation methods fine-tune the text-to-image model such that it generates images with the watermark intrinsically embedded.

Although these three types differ in watermark embedding, all require a watermark decoder during detection, denoted as  $Dec$ . We define the ground-truth watermark as  $w_t$ . During detection, the watermark decoder  $Dec$  extracts watermark  $w$  from a given image  $x$ , i.e.,  $w = Dec(x)$ . If the distance between the ground-truth watermark  $w_t$  and the decoded watermark  $w$  is below a threshold  $\tau$ , i.e.,  $d(w_t, w) < \tau$ , the image  $x$  is detected as watermarked and thus AI-generated, where  $d$  is a distance metric, e.g.,  $\ell_0$ -norm or  $\ell_1$ -norm. The threshold  $\tau$  can be selected to ensure a desired probability of falsely detecting a non-watermarked image as watermarked based on the assumption that  $w_t$  is sampled uniformly at random. Formally, a watermark-based detector  $D$  is defined as follows:

$$D(x) = \mathbb{I}(d(w_t, Dec(x)) < \tau), \quad (1)$$

where  $\mathbb{I}$  is the indicator function, i.e.,  $\mathbb{I}(d(w_t, Dec(x)) < \tau)$  is 1 if  $d(w_t, Dec(x)) < \tau$  and 0 otherwise. Different watermarking methods use various ground-truth watermarks  $w_t$ , decoders  $Dec$ , and/or distance metrics  $d$ . **Pre-generation watermark:** Given a prompt and a seed, a typical text-to-image model iteratively denoises the seed as an image. Pre-generation watermarks [17, 21] embed a watermark into an AI-generated image via encoding a signal into the seed. We use the state-of-the-art pre-generation watermarking method, Tree-Ring [17], in our experiments.

- **Tree-Ring [17].** When generating an image using a text-to-image model, Tree-Ring selects a seed  $x_T$  whose Fourier transform contains a specific pattern, which serves as the ground-truth watermark, i.e.,  $w_t = FT(x_T)$ , where  $FT$  means Fourier transform. For detection, given an image  $x$ , a seed is reconstructed from it using inverse

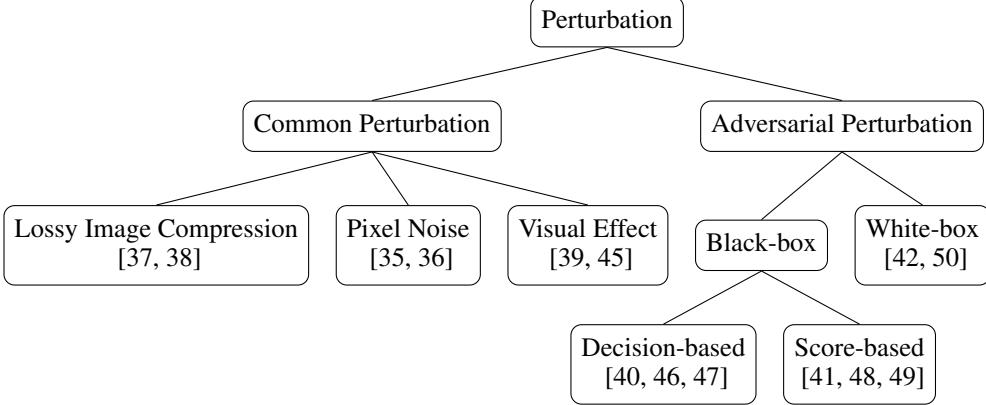


Figure 4: Our taxonomy of perturbations to deceive a detector.

DDIM sampling [44]. Then, a watermark is extracted from the reconstructed seed. The decoded watermark can be expressed as  $Dec(x) = FT(iDDIM(x))$ , where  $iDDIM$  is the inverse DDIM process. If the  $\ell_1$ -norm distance between the decoded watermark and the ground-truth one is below the threshold  $\tau$ , the image  $x$  is classified as watermarked and thus AI-generated.

**Post-generation watermark:** In a post-generation watermarking method [15, 16, 20], the ground-truth watermark  $w_t$  is a bitstring. These methods employ a watermark encoder,  $Enc$ , to embed the watermark into an image by subtly altering the image’s pixel values, producing a watermarked image. In our experiments, we consider the state-of-the-art post-generation watermarking methods: StegaStamp [15] and its smoothed version [16].

- **StegaStamp [15].** StegaStamp uses neural networks as the watermark encoder  $Enc$  and the decoder  $Dec$ . The two neural networks can be jointly trained using an image dataset. For detection, given an image  $x$ , if the  $\ell_0$ -norm distance (i.e., the number of mismatched bits) between the ground-truth watermark  $w_t$  and the decoded watermark  $Dec(x)$  is smaller than a threshold  $\tau$ , the image is classified as AI-generated.
- **Smoothed StegaStamp [16].** Smoothed StegaStamp is an enhanced version of StegaStamp, which provides certified robustness guarantees against bounded perturbations. During detection, given an image  $x$ ,  $N$  random Gaussian noises are added to it to construct  $N$  noisy images; the watermark decoder  $Dec$  extracts a watermark from each noisy image; and the  $\ell_0$ -norm distance between each decoded watermark and the ground-truth watermark  $w_t$  is calculated. Finally, the image  $x$  is classified as AI-generated if the median of these  $N$   $\ell_0$ -norm distances is smaller than the threshold  $\tau$ . Smoothed StegaStamp guarantees that the detection result is unaffected by any  $\ell_2$ -norm bounded perturbation added to  $x$ .

**In-generation watermark:** In-generation watermarks [18, 19] modify the parameters of a text-to-image model, ensuring that the generated images contain a watermark. In our experiments, we use Stable Signature[18], state-of-the-art in-generation watermark developed by Meta.

- **Stable Signature [18].** Stable Signature is based on the post-generation watermarking method HiDDeN [20]. Given a HiDDeN watermarking decoder  $Dec$  and a ground-truth watermark  $w_t$ , Stable Signature fine-tunes the text-to-image model so that the watermark  $w_t$  can be extracted from any generated image  $\hat{x}$ , i.e.,  $Dec(\hat{x}) \approx w_t$ . For detection, an image  $x$  is classified as AI-generated if  $\ell_0(w_t, Dec(x)) < \tau$ .

### 3 Taxonomy of Perturbations

#### 3.1 Removal and Forgery Attacks

To deceive an image detector, an attacker can introduce perturbations to manipulate an AI-generated image to appear non-AI-generated (*removal attack*), or, conversely, make a non-AI-generated image appear AI-generated (*forgery attack*). In a removal attack, the attacker adds perturbation  $\delta$  to an AI-generated image  $x_a$  so that the detector  $D$  mistakes it for a non-AI-generated image, i.e.,  $D(x_a + \delta) = 0$ . In a forgery attack, the attacker adds perturbation  $\delta$  to a non-AI-generated image  $x_h$  so that the detector  $D$  mistakes it for an AI-generated image, i.e.,  $D(x_h + \delta) = 1$ .

To execute these attacks, the attacker can apply either a *common perturbation* or *adversarial perturbation* to alter the AI-generated or non-AI-generated images. Common perturbation involves using standard image processing techniques, whereas adversarial perturbation introduces carefully crafted noise to an image. Figure 4 shows the detailed

taxonomy of perturbations that may be introduced to deceive a detector. It is important to note that common perturbations can also be applied by normal users for benign purposes, and may not always correspond to an attack.

### 3.2 Common Perturbation

These perturbations arise from common image processing operations. We categorize them into three types: lossy image compression [37, 38], pixel noise [35, 36], and visual effect [39, 45]. Each type of perturbation has a parameter, which controls the amount of perturbation added to an image.

**Lossy image compression:** Lossy image compression aims to reduce image file sizes but introduces compression artifacts that can degrade image quality. We incorporate two widely used image compression methods [37, 38].

- **JPEG Compression [37].** This method uses a quality factor  $q$  to balance compression ratio and image quality. A smaller  $q$  results in more aggressive compression.
- **High Efficiency Image File Format (HEIF) [38].** HEIF also uses a quality factor  $q$  to balance image quality and file size. A smaller  $c$  results in more aggressive compression.

**Pixel noise:** Pixel noise perturbation introduces random noise at the pixel level.

- **Gaussian noise.** Gaussian noise is characterized by a Gaussian distribution with zero mean and standard deviation  $\sigma$ , which controls the noise intensity.
- **Rayleigh noise.** Rayleigh noise is characterized by Rayleigh distribution with a scale parameter  $\sigma$ , which also determines the intensity of the noise.

**Visual effect:** Visual effect introduces carefully designed modifications to pixel values.

- **Gaussian blur.** Gaussian blur reduces image sharpness by averaging pixel values within a local neighborhood, following a Gaussian distribution. The blur radius  $r$  determines the extent of the smoothing effect.
- **Brightness adjustment.** This method adjusts the brightness of an image by adding a specified brightness value  $b$  to the pixel intensities.
- **Contrast adjustment.** Contrast adjustment scales the difference between each pixel and the mean pixel value. Pixel intensities are centered by subtracting the mean, multiplied by a contrast factor  $c$ , and re-centered. A factor  $c > 1$  increases contrast, making an image more vivid, while  $0 < c < 1$  reduces contrast, flattening details.
- **Elastic blur.** Elastic blur applies a spatial distortion to an image by shifting pixel locations according to a smooth displacement field. The degree of distortion is controlled by a parameter  $\alpha$ , which defines the intensity of the transformation. Larger  $\alpha$  causes more pronounced warping, while smaller  $\alpha$  leads to subtler distortions.

### 3.3 Adversarial Perturbation

Adversarial perturbations are intentionally crafted by attackers to execute removal or forgery attacks aimed at deceiving a detector. Specifically, given an image  $x$ , an adversarial perturbation  $\delta$  is found to deceive the detector  $D$ , i.e.,  $D(x + \delta) \neq D(x)$ , where  $x$  is an AI-generated image in removal attacks and a non-AI-generated image in forgery attacks. These adversarial perturbations can be identified by adapting adversarial examples [50] to detectors. Specifically, we treat a detector as a binary classifier and apply adversarial examples to it. Based on different levels of attackers' background knowledge, we consider two attack settings: *black-box* and *white-box*.

**Black-box attacks:** In the black-box setting, we assume the attacker can query the detector  $D$  using various images. Given an image  $x$  and a query budget, these attacks iteratively query  $D$  to optimize  $\delta$  until deceiving the detector or reaching the query budget. Based on the detector's response, these attacks can be further categorized into *decision-based* [40, 46, 47] and *score-based* [41, 48, 49]. In decision-based attacks, the detector simply indicates whether the queried image is AI-generated or not. In score-based attacks, the detector not only provides this detection result but also returns additional information, such as classification probability (for passive detectors) or distance between the decoded watermark and the ground-truth one (for watermark-based detectors). In our experiments, we use the HopSkipJump attack [40], a decision-based attack, and the Square attack [41], a score-based attack.

- **HopSkipJump (HSJ) attack [40].** Given an image  $x$ , this method begins with a large perturbation  $\delta$  that successfully deceives the detector  $D$ , i.e.,  $D(x + \delta) \neq D(x)$ . Such initial perturbation  $\delta$  can be obtained via applying severe common perturbations onto the image  $x$  [31]. The perturbation  $\delta$  is then iteratively optimized using gradients estimated from the detector's responses. Specifically, the HopSkipJump attack defines a loss function to quantify the attack's effectiveness and uses Monte Carlo estimation [51] to approximate its gradient. This gradient is then used to search for the minimal perturbation capable of deceiving the detector.
- **Square attack [41].** This method employs randomly generated noise to iteratively perturb randomly selected square regions of an image  $x$ . Specifically, in each iteration, the attack applies noise to the selected square regions if the

Table 1: Summary of our datasets. Each dataset includes 11,000 non-AI-generated and 11,000 AI-generated images, with separate columns for training and testing splits.

Dataset	Non-AI-generated			AI-generated		
	Data source	#Training	#Testing	Model	#Training	#Testing
<b>MS</b>	MSCOCO	10,000	1,000	Stable Diffusion	10,000	1,000
<b>GD</b>	Google CC	10,000	1,000	DALL-E 3	10,000	1,000
<b>FD</b>	Flickr30k	10,000	1,000	DeepFloyd IF	10,000	1,000
<b>TH</b>	TextCaps	10,000	1,000	Hunyuan-DiT	10,000	1,000

noise reduces the classification probability of the correct class (for passive detectors) or the bitwise accuracy (for watermark-based detectors).

**White-box attacks:** In the white-box setting, we assume that the attacker has full knowledge of the detector  $D$ , including its parameters and architecture. Given an image  $x$  and a perturbation budget  $r$ , the attacker’s objective is to identify a perturbation  $\delta$  within the budget that is most likely to deceive the detector  $D$ . The process of finding this perturbation  $\delta$  can be formulated as an optimization problem, typically solved using gradient-based methods [42, 50].

For passive detectors, the attacker optimizes  $\delta$  such that the detector’s prediction is opposite to the original prediction. The optimization problem is defined as follows [42]:

$$\max_{\delta} CE(x + \delta, D(x); D) \quad s.t. \|\delta\|_{\infty} \leq r, \quad (2)$$

where  $CE$  denotes the cross-entropy loss of  $D$  for  $x + \delta$  when treating  $D(x)$  as its label, and  $\|\cdot\|_{\infty}$  indicates  $\ell_{\infty}$ -norm. For watermark-based detectors, since the decoder  $Dec$  does not output a class probability, the typical white-box adversarial example methods cannot be directly applied. Following Jiang et al. [31], the attacker optimizes a perturbation  $\delta$  to either maximize the distance between the ground-truth watermark  $w_t$  and decoded watermark  $Dec(x + \delta)$  for a removal attack or minimize it for a forgery attack. Specifically, for a removal attack, the optimization problem can be formulated as follows:

$$\max_{\delta} l_2(w_t, Dec(x_a + \delta)) \quad s.t. \|\delta\|_{\infty} \leq r, \quad (3)$$

where  $x_a$  is a watermarked, AI-generated image. For a forgery attack, the optimization problem is as follows:

$$\min_{\delta} l_2(w_t, Dec(x_h + \delta)) \quad s.t. \|\delta\|_{\infty} \leq r, \quad (4)$$

where  $x_h$  is a non-AI-generated image. To solve the optimization problems for both passive and watermark-based detectors, we employ *projected gradient descent (PGD)* [42] to optimize  $\delta$ . Specifically, we optimize  $\delta$  under an  $\ell_{\infty}$ -norm constraint defined by the perturbation budget  $r$ . If  $\delta$  exceeds this budget, it is projected back onto the  $\ell_{\infty}$ -norm ball.

## 4 Collecting Datasets

Our goal is to collect diverse image datasets with varied content generated by different text-to-image models. To achieve this, we compiled four datasets, each containing both non-AI-generated and AI-generated images. For each dataset, we followed these steps: beginning with a public dataset of non-AI-generated images with captions, we sampled 11,000 images. Then, we used the corresponding captions as prompts to generate 11,000 AI-generated images through a text-to-image model. Using the same captions as prompts helps minimize the influence of image content on the distinction between AI-generated and non-AI-generated images. We pair each dataset of non-AI-generated images with a different text-to-image model, though the pairings are not based on any specific pattern. Figure 14-17 in Appendix show examples of images from each dataset. Table 1 provides an overview of the four datasets, which we elaborate on in the following.

**MSCOCO – Stable Diffusion (MS):** MSCOCO [52] is a dataset containing 320,000 non-AI-generated images with captions, featuring common scenes intended for object detection, segmentation, and captioning tasks. We sampled 11,000 non-AI-generated images from MSCOCO. Each image includes one or more captions; for each, we selected the first caption as a prompt to generate an AI-generated image using Stable Diffusion v2.1 [1]. These prompts have an average of 10.2 tokens, with a standard deviation of 2.4 tokens, and a median of 10 tokens.

**Google CC – DALL-E 3 (GD):** The Google CC dataset [53] contains more than 3,300,000 non-AI-generated images sourced from various websites, where the image captions are extracted and refined from the associated HTML tags.

We sampled 11,000 non-AI-generated images from Google CC. Furthermore, for each non-AI-generated image, we treat its caption as a prompt to generate an AI-generated image using DALL-E 3 (version 2024/02/01). These prompts have an average of 6.9 tokens, with a standard deviation of 1.5 tokens, and a median of 6 tokens.

**Flickr30k – DeepFloyd IF (FD):** The Flickr30k dataset [54] contains more than 30,000 non-AI-generated images, each paired with multiple captions. We sampled 11,000 non-AI-generated images from Flickr30k. Similar to our treatment of MSCOCO, for each image, we use its first caption as a prompt to generate an AI-generated image through DeepFloyd IF [3]. We use the first two modules of DeepFloyd IF (“IF-I-XL-v1.0” and “IF-II-L-v1.0”) in its pipeline. These prompts have an average of 17.3 tokens, with a standard deviation of 5.5 tokens, and a median of 16 tokens.

**TextCaps – Hunyuan-DiT (TH):** The TextCaps dataset [55] includes 39,408 non-AI-generated images and there are 5 captions for each image. We sampled 11,000 non-AI-generated images from TextCaps. Furthermore, for each image, we use its first caption as a prompt to generate an AI-generated image using Hunyuan-DiT [4]. These prompts have an average of 12.6 tokens, with a standard deviation of 3.7 tokens, and a median of 12 tokens.

**Training and testing splits:** For each dataset, we sample 10,000 non-AI-generated and 10,000 AI-generated images for the training set, reserving the remaining 1,000 non-AI-generated and 1,000 AI-generated images for the testing set. For passive detectors, we train them on the training set of each dataset and test them on the corresponding testing set. For watermark-based detectors, we train the watermark encoders and decoders using a dataset separate from the four primary datasets, with further details provided in Section 5.2. It is worth noting that our training setup gives advantages to passive detectors.

## 5 Training Detectors

### 5.1 Passive Detectors

We evaluate three passive detectors: UnivCLIP [9], FreqDetect [10], and DE-FAKE [8]. We use the publicly available code for each detector. Unless specified otherwise, for each of our four datasets, the training set is used to train these detectors, and their performance is evaluated using the corresponding testing set. Detailed training procedures for each detector are outlined below.

**UnivCLIP:** UnivCLIP employs the ViT-B/32 CLIP model [56] to extract spatial features from images, with a linear layer serving as the classifier. We train the classifier using the Adam optimizer, with a batch size of 500 for 800 iterations. The initial learning rate is set at  $3 \times 10^{-4}$  and decays to  $1 \times 10^{-6}$  following a cosine decay schedule. To enhance robustness, we use *training with perturbations*, where images are perturbed during training. These perturbations include JPEG compression, Gaussian noise, Gaussian blur, brightness, and contrast adjustments. Each image undergoes a perturbation picked from the five choices uniformly at random in each iteration.

**FreqDetect:** FreqDetect uses the DCT frequency coefficients as image features, with logistic regression serving as the classifier. The logistic regression classifier is trained using the L-BFGS solver over 1,000 iterations with an  $\ell_2$ -norm penalty. We also apply training with perturbations to enhance robustness.

**DE-FAKE:** DE-FAKE uses the ViT-B/32 CLIP model [56] to extract features from both images and captions, with a three-layer multilayer perceptron serving as the classifier based on these features. Since DE-FAKE takes image-caption pairs as input, the training set includes both images and their corresponding captions. During training, the classifier is optimized using the Adam optimizer with a batch size of 500 for 800 iterations. The learning rate is initially set to  $3 \times 10^{-4}$  and decays to  $1 \times 10^{-6}$  following a cosine decay schedule. Additionally, training with perturbations is used to enhance robustness. For testing, we use BLIP [43] to generate a caption for each image and classify the resulting image-caption pair with the trained classifier.

### 5.2 Watermark-based Detectors

In our experiments, we use four watermark-based detectors: Tree-Ring [17], StegaStamp [15], Smoothed StegaStamp [16], and Stable Signature [18]. Their respective training details (if any) are provided below.

**Tree-Ring:** Tree-Ring does not require training, and we use the default watermark embedding and detection settings provided by Wen et al. [17]. Specifically, a circle pattern with a radius of 10 serves as the ground-truth watermark  $w_t$ . For detection, Tree-Ring utilizes the inverse DDIM process to reconstruct a seed for a given image. This reconstruction process requires a prompt, as is typical for text-to-image models. In our experiments, we follow Wen et al. [17] to use an empty string as the prompt. Note that Tree-Ring is tailored to Stable Diffusion, and thus we will only use it to watermark AI-generated images in our MS dataset.

**StegaStamp:** For StegaStamp, we train the watermark encoder and decoder on 10,000 non-AI-generated images sampled from the MSCOCO dataset [52]. Importantly, these images do not overlap with those in our MS dataset, which is to show the generalization ability across datasets.

Following Tancik et al. [15], we employ a U-Net architecture as the watermark encoder  $Enc$  and a spatial transformer network [57] as the watermark decoder  $Dec$ . The watermark is a bitstring with 32 bits. The training process involves two phases: *standard training* and *training with perturbations*. In the standard training phase, the watermark encoder and decoder are jointly trained such that 1) the watermark decoded from a watermarked image is similar to the true watermark, i.e.,  $Dec(Enc(x, w)) \approx w$  for any watermark  $w$ , where  $Enc(x, w)$  indicates a watermarked image with watermark  $w$ ; and 2) a watermarked image (i.e.,  $Enc(x, w)$ ) looks visually the same as the image  $x$ .

Training with perturbations differs by applying perturbations to the watermarked images during training. This encourages the watermark encoder to embed a robust pixel pattern into an image as a watermark. Specifically, the watermark encoder and decoder are jointly trained such that the watermark decoded from a perturbed watermarked image is similar to the true watermark, i.e.,  $Dec(P(Enc(x, w))) \approx w$  for  $w$ , where  $P$  is a perturbation operation. We apply the same five types of perturbations as training the passive detectors. For each training image in each iteration, we randomly sample one perturbation from these five options and apply it to the corresponding watermarked image.

We train the watermark encoder and decoder for 3,000 iterations with a batch size of 32. The first 1,000 iterations use standard training, while the remaining 2,000 iterations employ training with perturbations to enhance robustness. After training, a random 32-bit bitstring is sampled as  $w_t$ , which is embedded into AI-generated images.

**Smoothed StegaStamp:** Smoothed StegaStamp uses the trained StegaStamp model, so it does not require extra training. Specifically, Smoothed StegaStamp applies the regression smoothing [16] to the trained StegaStamp model. During detection, given an image  $x$ , Smoothed StegaStamp samples 100 Gaussian noise vectors with zero mean and a standard deviation of 0.1, adds them to the image to construct 100 noisy images. For each noisy image, we use the StegaStamp’s watermark decoder  $Dec$  to decode a watermark and calculate its  $\ell_0$ -norm distance with the ground-truth watermark  $w_t$ . We then take the median across the 100  $\ell_0$ -norm distances as the final  $\ell_0$ -norm distance, and classify the image  $x$  as AI-generated if the final  $\ell_0$ -norm distance is smaller than a threshold  $\tau$ .

**Stable Signature:** For Stable Signature, we use the publicly available text-to-image model [18]. In this model, the ground-truth watermark  $w_t$  is a 48-bit bitstring. This model is adversarially trained with JPEG compression, random cropping, and random resizing. The watermark decoder  $Dec$  follows the architecture of HiDDeN [20]. We note that the publicly available text-to-image model fine-tuned by Stable Signature is Stable Diffusion, and thus in experiments, we will only apply Stable Signature to watermark AI-generated images in our MS dataset.

**Detection threshold  $\tau$ :** Passive detectors rely on standard classifiers with widely used detection thresholds. In contrast, via properly setting the threshold  $\tau$ , watermark-based detectors can ensure a desired probability of falsely detecting a non-AI-generated image as AI-generated, assuming the ground-truth watermark is sampled uniformly at random. Following previous studies [31], we select  $\tau$  in our experiments to ensure this probability is less than 0.0001. Specifically, for StegaStamp and Smoothed StegaStamp,  $\tau$  is set to 0.8125; for Stable Signature,  $\tau$  is set to 0.7708; and for Tree-Ring,  $\tau$  is set to 70.1875. It is important to note that this probability differs from the false positive rate (FPR) reported in our experiments. FPR refers to the fraction of non-AI-generated images falsely detected as AI-generated for a given ground-truth watermark.

## 6 Benchmarking Results

### 6.1 Evaluation Metrics

**FNR, FPR, and ACC:** We evaluate the performance of the detectors using *false negative rate (FNR)*, *false positive rate (FPR)*, and *accuracy (ACC)*. FNR is the fraction of AI-generated images incorrectly detected as non-AI-generated; FPR is the fraction of non-AI-generated images incorrectly detected as AI-generated; and ACC is the overall fraction of correct detection. Unless otherwise mentioned, for each of our four datasets, we calculate FNR, FPR, and ACC of a detector using the testing set comprising 1,000 AI-generated and 1,000 non-AI-generated images. Under removal (or forgery) attacks, we add common or adversarial perturbations to AI-generated (or non-AI-generated) images. A smaller FNR (or FPR) for the perturbed AI-generated (or non-AI-generated) images indicates that the detector is more robust against removal (or forgery) attacks.

**LPIPS:** We will apply common and adversarial perturbations to the images to evaluate the robustness of detectors. In those experiments, we further assess the visual similarity between an image and its perturbed version using the *Learned Perceptual Image Patch Similarity (LPIPS)* [58]. LPIPS is a widely used metric that quantifies the visual similarity between two images by comparing their deep feature representations. In our experiments, the feature representation of

Table 2: FNRs, FPRs, and ACCs of various detectors on our four datasets. We cannot get FNRs (and thus ACCs) for Tree-Ring and Stable Signature on GD, FD and TH datasets since their implementations are only applicable for Stable Diffusion model.

Detector		MS			GD			FD			TH		
		FNR	FPR	ACC									
Passive	UnivCLIP	0.05	0.07	0.94	0.01	0.06	0.96	0.01	0.01	0.99	0.02	0.09	0.94
	FreqDetect	0.26	0.15	0.80	0.18	0.28	0.77	0.08	0.06	0.93	0.18	0.30	0.76
	DE-FAKE	0.05	0.06	0.94	0.01	0.05	0.97	0.01	0.01	0.99	0.03	0.06	0.96
Watermark-based	Tree-Ring	0.00	0.00	1.00	-	0.00	-	-	0.00	-	-	0.00	-
	StegaStamp	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
	Smoothed StegaStamp	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
	Stable Signature	0.01	0.00	0.99	-	0.00	-	-	0.00	-	-	0.00	-

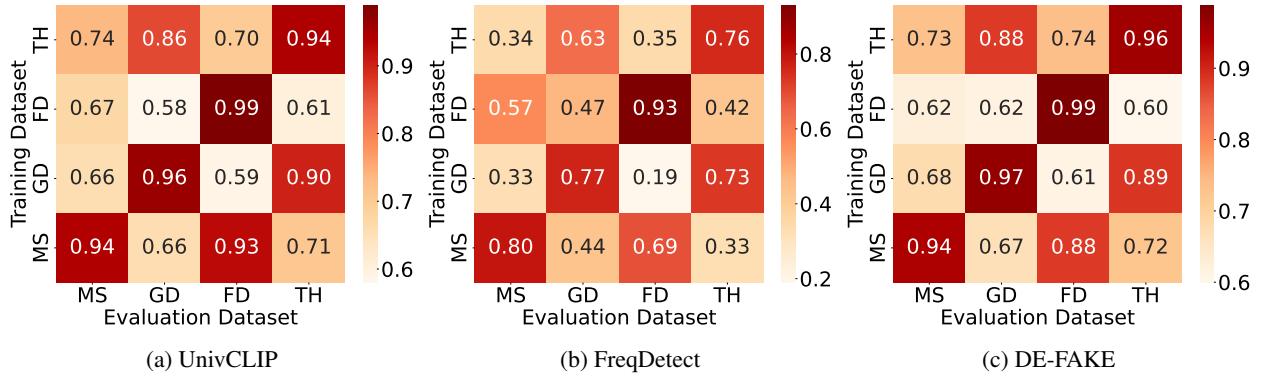


Figure 5: ACCs of passive detectors when trained and evaluated across datasets.

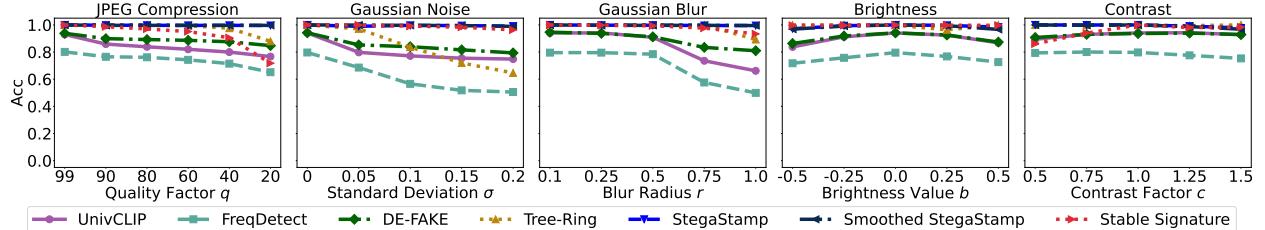


Figure 6: ACCs of various detectors on the MS dataset under 5 types of common perturbations seen during training. The FNR and FPR results are shown in Figure 18 in Appendix.

an image is calculated using the pre-trained VGG neural network [59]. A lower LPIPS score indicates that an image and its perturbed version are more visually similar.

**Running time:** We also evaluate the efficiency of the detectors by measuring their runtime per image detection. Given the high volume of images processed in real-world scenarios, detection may need to be performed frequently and at scale. For instance, social media platforms may label AI-generated images before publication. According to [60], X (formerly Twitter) publishes approximately 3,000 images per second. This immense volume highlights the necessity for detectors to operate efficiently.

## 6.2 No Perturbations

**In-distribution performance:** Table 2 presents the FNRs, FPRs, and ACCs of various detectors for each dataset when no perturbations are added to the testing images. It is important to note that all passive detectors are trained and evaluated respectively on the training and testing set of each dataset, i.e., the testing and training images follow the same distribution. In contrast, watermark-based detectors are trained using a dataset that is different from the four datasets (see Section 5.2 for more details). This setting gives advantages to passive detectors. We also note that the public implementations of Tree-Ring and Stable Signature are only applicable to watermark images generated by Stable Diffusion. Therefore, we can only calculate FNR (and thus ACC) on the MS dataset. However, we can calculate their FPRs on all the four datasets.

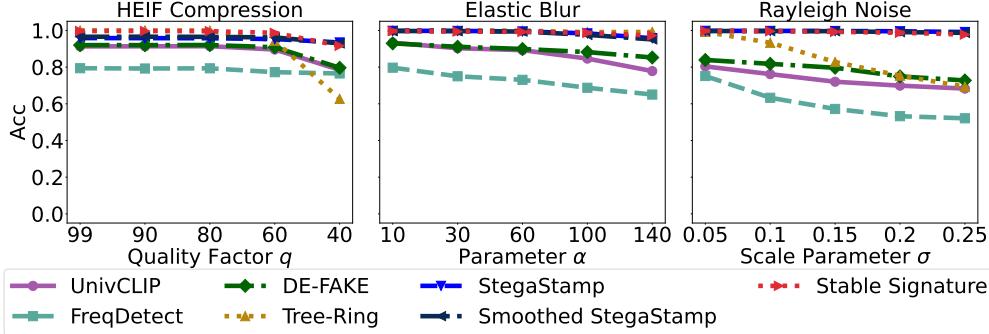


Figure 7: ACCs of various detectors under 3 types of common perturbations unseen during training. The FNR and FPR results are shown in Figure 13 in Appendix.

First, we observe that all watermark-based detectors outperform passive detectors. Specifically, all watermark-based detectors except Stable Signature achieve an ACC of 1.00, while Stable Signature achieves an ACC of 0.99 on MS. In contrast, the best-performing passive detector achieves an average ACC of only 0.965 across the four datasets, even when trained and evaluated on data from the same distribution. This highlights both the superior detection accuracy and the image distribution insensitivity of watermark-based detectors.

Second, among the passive detectors, DE-FAKE demonstrates the highest performance, achieving an average ACC of 0.965, while the other two passive detectors achieve average ACCs of 0.958 and 0.815, respectively. DE-FAKE’s superior performance may result from its use of both images and their corresponding captions for detection, as the additional information likely enhances detection capabilities.

Third, the performance of passive detectors varies significantly across datasets. For instance, FreqDetect achieves an ACC of 0.93 on FD but only 0.76 on TH, suggesting that passive detectors may be sensitive to image distribution.

**Out-of-distribution performance:** In real-world scenarios, the distribution of testing data may differ from the training data. For instance, a text-to-image model may be updated, or a detector might be required to classify non-AI-generated images from a different distribution or AI-generated images from different models. Thus, we further evaluate the out-of-distribution performance of passive detectors by training on one dataset and testing on others. Since Table 2 already presents the out-of-distribution performance of watermark-based detectors, we focus our analysis here on passive detectors. Specifically, we train each passive detector using the training set of one dataset and evaluate it on the testing set of each of the four datasets. Figure 5 shows the ACC of the three passive detectors under these cross-dataset conditions. Each grid cell shows the ACC obtained when a detector is trained on one dataset and evaluated on another.

We observe that the diagonal cells (from the bottom left to the top right), which represent detectors trained and evaluated on the same dataset, consistently outperform the off-diagonal cells. For example, UnivCLIP achieves an ACC of 0.99 on FD when training and testing on the same dataset, but only achieves 0.58 when trained on FD and tested on GD. Similarly, DE-FAKE achieves an ACC of 0.94 on MS when evaluated in-distribution, but this drops to 0.67 when tested on GD. Overall, the average out-of-distribution ACC across the off-diagonal cells is 29% lower than the average in-distribution ACC, indicating a substantial performance drop when tested on unseen distributions. This finding suggests that passive detectors struggle with out-of-distribution generalization. A likely reason is that passive detectors primarily rely on memorizing content-specific features, making them less adaptable to new, unseen image characteristics present in different datasets.

**Visual quality of watermarked images:** To assess the visual quality of watermarked images, we calculate the LPIPS between each AI-generated image and its watermarked counterpart. The results are presented in Figure 12 in Appendix. Our results show that watermarks preserve the visual quality of AI-generated images.

### 6.3 Common Perturbations

To evaluate a detector’s robustness on common perturbation, we calculate FNR, FPR, and ACC for each type of perturbation. Each perturbation type includes a parameter to control its intensity; for example, JPEG compression uses a quality factor  $q$ . Given a specific perturbation type and parameter value, we apply the perturbation to the 1,000 testing AI-generated images in a dataset as a removal attack and compute the detector’s FNR on these perturbed images. We also apply the same perturbation to the 1,000 testing non-AI-generated images as a forgery attack and compute the detector’s FPR. Subsequently, we calculate ACC based on the combined 2,000 perturbed images. Additionally, for each parameter value, we measure the LPIPS between an image and its perturbed version, averaging this score across

all 2,000 images. This process is repeated across different parameter values for each perturbation type to assess the detector’s robustness, provided that the perturbations do not significantly degrade image quality (i.e., LPIPS remains below a certain threshold).

**Performance under seen common perturbations:** Figure 6 shows ACCs of various detectors on the MS dataset under five types of common perturbations with varying parameter values; Figure 18 in Appendix shows the FNRs and FPRs; and Figure 19 in Appendix shows the LPIPS for each perturbation type. All detectors except Tree-Ring have undergone training with perturbations on these five types of perturbations, and thus they are referred to as *seen common perturbations*. Training with perturbations is not applicable to Tree-Ring because it does not require training.

First, we observe that StegaStamp and Smoothed StegaStamp consistently outperform passive detectors across all tested perturbations. For example, under JPEG compression with a quality factor of  $q = 40$ , Smoothed StegaStamp achieves an ACC of 1.00, whereas the top-performing passive detector, DE-FAKE, only reaches 0.88. Other watermark-based detectors also demonstrate superior performance over passive detectors in most cases. Specifically, common perturbations fail to conduct successful forgery attacks against any watermark-based detectors (i.e., FPRs remain close to 0). While such common perturbations also fail to conduct successful removal attacks against StegaStamp and Smoothed StegaStamp (i.e., FNRs remain close to 0), some of them (e.g., JPEG compression with a quality factor  $q = 20$ ) can perform successful removal attacks on Tree-Ring and Stable Signature. In contrast, passive detectors are vulnerable to both removal and forgery attacks under common perturbations. This superior robustness of watermark-based detectors is likely due to the stability provided by the embedded watermarks.

Second, we observe that Smoothed StegaStamp stands out as the most robust watermark-based detector. It consistently achieves slightly lower FNRs and FPRs, as well as higher ACCs compared to StegaStamp. Moreover, it significantly outperforms Tree-Ring and Stable Signature in some scenarios, particularly under more aggressive common perturbations. For instance, under JPEG compression with a quality factor  $q = 20$ , Smoothed StegaStamp achieves an ACC of 0.99, while Tree-Ring and Stable Signature reach ACCs of 0.88 and 0.72, respectively. This robustness advantage likely stems from its use of the randomized smoothing technique [61]. Notably, Smoothed StegaStamp offers certified robustness against bounded perturbations. Specifically, its detection result for an image remains provably unaffected by any perturbation whose  $\ell_2$ -norm is within a certain bound.

Third, among passive detectors, we observe that DE-FAKE provides the best performance under seen common perturbations. While passive detectors are generally less robust, DE-FAKE consistently achieves higher ACC than UnivCLIP and FreqDetect. For example, under Gaussian blur with a radius  $r = 1.0$ , DE-FAKE reaches an ACC of 0.81, compared to UnivCLIP’s 0.66.

**Performance under unseen common perturbations:** In real-world scenarios, images may encounter perturbations, referred to as *unseen common perturbations*, that were not included in a detector’s training with perturbations. To assess the robustness of different detectors to such unseen common perturbations, we further evaluate their performance under three other types of common perturbations: HEIF, Rayleigh noise, and elastic blur. Figure 7 presents ACCs of various detectors on the MS dataset under these unseen common perturbations with varying parameter values, while Figure 13 in Appendix shows the FNRs and FPRs.

Notably, the key observations regarding robustness under seen common perturbations also hold for unseen ones. For example, watermark-based detectors continue to demonstrate strong robustness even with perturbation types not included in their training. Specifically, they maintain ACCs around 0.95 when the LPIPS is at most 0.20, whereas passive detectors achieve ACCs around (with mean of) 0.85. This result indicates that watermark-based detectors can reliably identify AI-generated images even under unseen common perturbations, underscoring their robustness.

**Training with vs. without perturbations:** By default, we train the detectors (except Tree-Ring) using 5 types of common perturbations. We observe that detectors trained without perturbations show reduced robustness against perturbations. For example, as shown in Figure 8, ACC of Smoothed StegaStamp on the MS dataset declines under JPEG compression across various quality factors when training is conducted without perturbations. These results highlight that training with perturbations enhances robustness, as ACCs are notably higher for detectors trained with them.

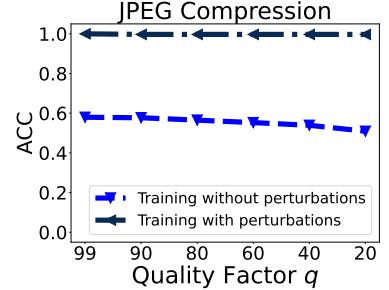


Figure 8: ACC of Smoothed StegaStamp on the MS dataset under JPEG compression with different quality factors when training with vs. without perturbations.

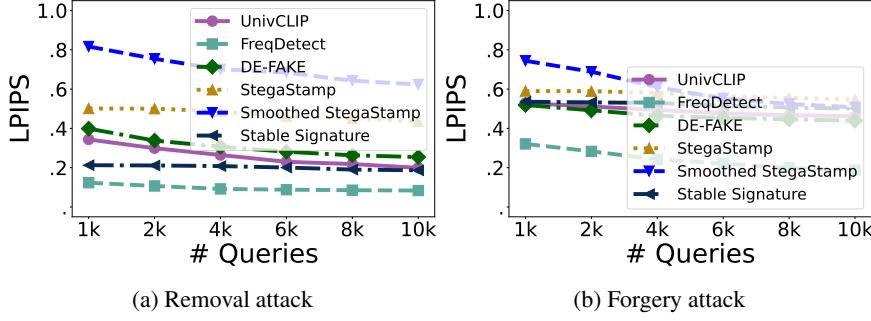


Figure 9: LPIPS for various detectors under removal and forgery attacks involving the black-box HopSkipJump attack with varying query budget.

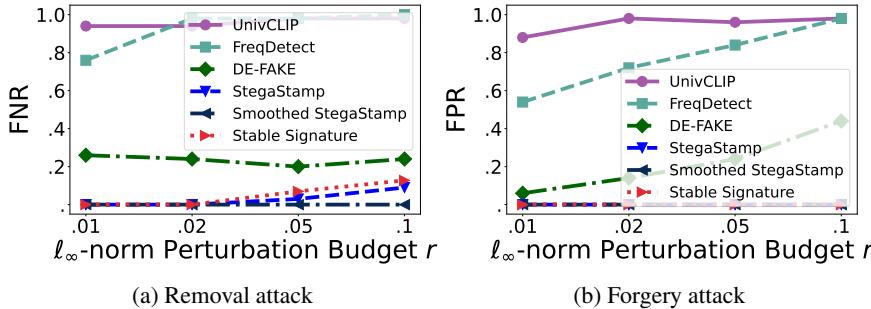


Figure 10: FNRs and FPRs of various detectors under removal and forgery attacks involving black-box Square attack with varying  $\ell_\infty$ -norm perturbation budgets  $r$ .

## 6.4 Adversarial Perturbations

Due to the inefficiency of Tree-Ring (details in Section 6.5), we exclude it in our evaluation on black-box attacks since they require a large number of queries to a detector for each image. Moreover, the inverse DDIM process in Tree-Ring causes gradients to accumulate, demanding excessive GPU memory to complete white-box attacks. Thus, we also exclude Tree-Ring from the evaluation on white-box attacks due to our limited computation resources.

**Black-box:** We evaluate removal and forgery attacks using HopSkipJump and Square attacks. For removal attacks using HopSkipJump on an AI-generated image, we initialize the attack with a JPEG-compressed version of the image that is classified as non-AI-generated. For forgery attacks on a non-AI-generated image, we initialize the attack with a watermarked, AI-generated image, as common perturbations like JPEG compression cannot forge a watermark. HopSkipJump ensures deceiving the detector during the attack process, maintaining FNRs and FPRs at 1. Therefore, we use the LPIPS between images and their final perturbed versions to measure the robustness of the detectors. Figure 9 presents the LPIPS values of various detectors under HopSkipJump attacks across different query budgets.

Given an  $\ell_\infty$ -norm perturbation budget  $r$ , the Square attack iteratively perturbs an image to deceive a detector. Unlike HopSkipJump, the Square attack does not guarantee deceiving a detector. Thus, to evaluate the robustness of a detector, we report its FNR and FPR under removal and forgery attacks, respectively, for varying  $\ell_\infty$ -norm perturbation budgets. Figure 10 shows the FNRs and FPRs of various detectors under Square attacks with different perturbation budgets. For each perturbation budget and image, the Square attack is allowed to query the detector up to 10,000 times in our experiments.

We observe that watermark-based detectors, particularly Smoothed StegaStamp, demonstrate strong robustness against black-box adversarial perturbations. Notably, forgery attacks consistently fail against watermark-based detectors. For instance, forgery attacks using HopSkipJump significantly degrade the visual quality of non-AI-generated images, resulting in large LPIPS values (Figure 20 in Appendix shows several examples). Similarly, forgery attacks using the Square attack achieve FPRs close to 0 against watermark-based detectors. In addition, Smoothed StegaStamp shows high resilience to removal attacks. Specifically, these attacks result in large LPIPS values (for HopSkipJump) or low FNRs (for Square attack). While other watermark-based detectors may eventually generate perturbed images with acceptable visual quality under high query budgets in HopSkipJump attacks, or show slight increases in FNRs under Square attacks, they generally maintain robust detection performance against these adversarial perturbations.

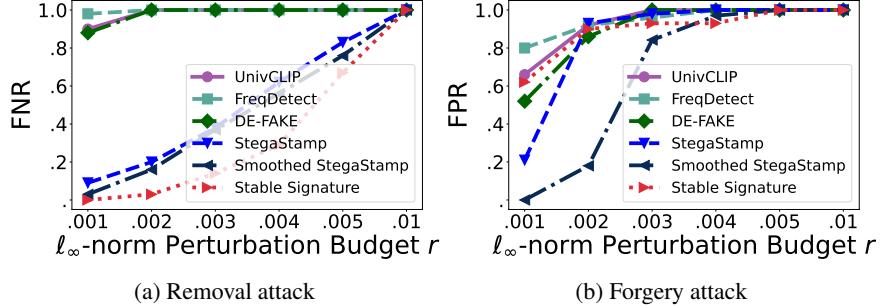


Figure 11: FNRs and FPRs of various detectors under removal and forgery attacks involving white-box adversarial perturbations with varying  $\ell_\infty$ -norm perturbation budgets  $r$ .

Table 3: Time (seconds) taken to classify one image for passive and watermark-based detectors.

Detector		Time
Passive	UnivCLIP	$6.48 \cdot 10^{-3}$
	FreqDetect	$6.71 \cdot 10^{-4}$
	DE-FAKE	$3.24 \cdot 10^{-1}$
Watermark-based	Tree-Ring	$2.17 \cdot 10^0$
	StegaStamp	$6.69 \cdot 10^{-3}$
	Smoothed StegaStamp	$6.79 \cdot 10^{-3}$
	Stable Signature	$1.39 \cdot 10^{-2}$

In contrast, passive detectors show limited robustness against black-box adversarial perturbations. Notably, DE-FAKE, the most robust passive detector, is effectively deceived by the HopSkipJump attack, resulting in low LPIPS values. Similarly, under the Square attack, DE-FAKE reaches an FNR greater than 0.2, and its FPR increases rapidly as the perturbation budget grows.

**White-box:** To evaluate the worst-case robustness of passive and watermark-based detectors, we apply white-box adversarial perturbations to execute both removal and forgery attacks. Specifically, given a detector, we find a white-box adversarial perturbation for each testing AI-generated (or non-AI-generated) image using PGD as a removal (or forgery) attack. For Smoothed StegaStamp, because the standard white-box attack in Section 3.3 is suboptimal, we adopt the white-box attack [16] tailored to Smoothed StegaStamp. Then, we compute the detector’s FNR (or FPR) of the perturbed AI-generated (or non-AI-generated) images. Figure 11 presents the FNRs and FPRs of various detectors on the MS dataset under white-box adversarial perturbations at different  $\ell_\infty$ -norm perturbation budgets  $r$ .

We observe that none of the detectors are robust against white-box adversarial perturbations. Specifically, for removal attacks, the FNRs of all detectors reach 1.0 when  $r$  is only 0.01. Similarly, for forgery attacks, the FPRs of all detectors reach 1.0 when  $r$  is only 0.005. However, watermark-based detectors demonstrate better robustness compared to passive detectors. Specifically, it requires large perturbations to make FNRs and FPRs of watermark-based detectors reach 1, compared to passive detectors. For instance, DE-FAKE reaches an FNR and FPR of 1.0 at  $r = 0.003$  for both removal and forgery attacks. In contrast, Smoothed StegaStamp reaches an FNR and FPR of 1.0 at  $r = 0.01$  and  $r = 0.005$  for removal and forgery attacks, respectively. StegaStamp and Smoothed StegaStamp are overall the most robust detectors, which achieve comparable FNRs and FPRs across perturbation budgets  $r$ . Although Stable Signature achieves comparable FNRs with StegaStamp and Smoothed StegaStamp, its FPRs are much larger than those of StegaStamp and Smoothed StegaStamp when  $r$  is small.

## 6.5 Detection Efficiency

Table 3 shows the time for each detector to classify one image averaged over 100 images. The detection time is defined as the time taken from receiving an input image until outputting the binary detection result. All measurements are conducted on a single A6000 GPU with 24GB memory and an Intel(R) Xeon(R) Gold 6230 CPU on a local machine. Tree-Ring is 1-4 orders of magnitude slower than other detectors. This is because it inverts the diffusion process during detection. DE-FAKE is the second slowest detector as it uses BLIP to generate a caption for each classification. FreqDetect is the most efficient detector due to its simplicity. UnivCLIP and StegaStamp have comparable efficiency and are the second most efficient.

Table 4: FNPs, FPRs, and ACCs of commercial passive detectors and watermark-based detectors on 100 AI-generated images and 100 non-AI-generated images under no perturbations and Gaussian noise perturbations.

Detector	No perturbation			Gaussian noise		
	FNP	FPR	ACC	FNP	FPR	ACC
HIVE	0.00	0.00	1.00	0.23	0.00	0.88
ILLUMINARTY	0.20	0.20	0.80	0.86	0.00	0.57
StegaStamp	0.00	0.00	1.00	0.00	0.00	1.00
Smoothed StegaStamp	0.00	0.00	1.00	0.00	0.00	1.00

## 6.6 Commercial Passive Detectors

We evaluate two commercial detectors: HIVE [13] and ILLUMINARTY [14], deployed by [hive.ai](#) and [illuminarty.ai](#), respectively. Although the specific algorithms of these commercial detectors are not publicly disclosed, they are considered passive because they detect AI-generated images without relying on watermarks embedded during image generation. HIVE is the state-of-the-art commercial detector according to a recent study [27]. Due to a daily limit of 100 queries and the need to manually upload images to the web interfaces,\* we sample 100 AI-generated images and 100 non-AI-generated images from the testing set of the MS dataset and use the commercial detectors to classify them. Additionally, we add Gaussian noise with a standard deviation of 0.2 to each image and then query the commercial detectors again to classify the perturbed images.

Table 4 presents the FNP, FPR, and ACC results for the two commercial passive detectors alongside two watermark-based detectors. In the scenario of no perturbations, HIVE and the two watermark-based detectors perform very well, while ILLUMINARTY does not. Specifically, HIVE achieves an ACC of 1.0. The superior performance of HIVE is consistent with prior findings on detecting AI-generated art images reported by Ha et al. [27]. In contrast, ILLUMINARTY shows a lower ACC of 0.8. This difference may be because HIVE’s training dataset likely includes a large number of AI images generated by a diverse set of text-to-image models and non-AI-generated images sourced from the Internet [62]. Since our dataset also comprises images collected from the Internet or generated using publicly accessible text-to-image models, there is a possibility that HIVE has encountered similar examples during training, thus boosting its performance on these images.

When Gaussian noise is added to each image, the performance gap between the detectors widens considerably. HIVE’s ACC decreases to 0.88, while ILLUMINARTY’s ACC drops to 0.57, indicating their vulnerability to common perturbations. In contrast, the watermark-based detectors maintain ACCs of 1.0 even under these perturbations, showing their superior robustness.

## 6.7 Summary and Recommendations

**Summary of findings:** Our comprehensive benchmark study reveals several key findings. Watermark-based detectors consistently outperform passive detectors across all conditions, including scenarios with no perturbations, common perturbations, and adversarial perturbations. Additionally, watermark-based detectors exhibit a remarkable insensitivity to variations in image distributions, in contrast to passive detectors, which are significantly affected. The state-of-the-art watermark-based detector, Smoothed StegaStamp, demonstrates strong robustness against both common and black-box perturbations. However, neither passive nor watermark-based detectors show robust performance under white-box adversarial perturbations, although watermark-based detectors still show better resilience in these cases. Finally, Smoothed StegaStamp, the best-performing watermark-based detector, is two orders of magnitude more efficient than DE-FAKE, the best-performing passive detector.

**Recommendations:** Based on our findings, we offer several recommendations. When both passive and watermark-based detectors are applicable, watermark-based detectors—particularly Smoothed StegaStamp—should be the preferred choice due to their superior effectiveness, robustness, and efficiency. Furthermore, training with perturbations should be adopted to enhance the detectors’ robustness.

We acknowledge that watermark-based detectors are not applicable for detecting AI images that have already been generated and propagated on the Internet before watermarks are deployed. Additionally, malicious actors may fine-tune public text-to-image models and use them to generate images without embedding watermarks. In these situations, passive detectors are necessary. Our study yields several recommendations for training effective passive detectors. First, training should include AI-generated images from a diverse set of models and non-AI-generated images from diverse sources to capture subtle differences in texture, color distribution, and structural characteristics typical of AI-

\*They offer APIs for a fee. We reached out to HIVE for API access and pricing information but did not receive a response.

generated images. Second, training with perturbations is beneficial, as it exposes the detectors to challenging cases and enhances robustness against perturbations that may encounter in real-world scenarios.

## 7 Discussion and Limitations

**Other modalities (text, audio, and video):** Similar studies can be conducted for other modalities, since both passive and watermark-based detectors have been developed for them. For example, in the context of AI-generated text, multiple passive detectors [63–65] and watermark-based detectors [66–68] have been proposed. For AI-generated audio, various passive [69, 70] and watermark-based detectors [71–73] have been developed as well. We believe our key take-away message that watermark-based detectors outperform passive detectors still hold, though it is a valuable future work to conduct a systematic benchmark study to confirm.

**Adversarial training:** In our experiments, we used training with perturbations rather than adversarial training [42]. The key difference lies in the approach to introducing perturbations during training. Adversarial training involves applying adversarial perturbations to each image, where the adversarial perturbations are specifically optimized during each training iteration to maximize the detector’s vulnerability. By contrast, training with perturbations involves randomly sampling a common perturbation (e.g., Gaussian noise) and applying it to each image in each training iteration. Although adversarial training may further improve robustness against adversarial perturbations, we opted not to use it in our experiments due to its significantly higher computational cost. Adversarial training requires additional optimization steps for each training image in each training iteration, making it less efficient and challenging to scale. Additionally, while adversarial training can improve robustness against white-box adversarial perturbations [31], i.e., a larger adversarial perturbation is needed to deceive a detector, it is still insufficient, i.e., the adversarial perturbation is still small.

## 8 Conclusion and Future Work

Through a comprehensive benchmark study comparing the effectiveness, robustness, and efficiency of passive and watermark-based detectors, we find that watermark-based detectors consistently outperform passive detectors in all scenarios with respect to effectiveness and robustness. Moreover, the most effective and robust watermark-based detector is two orders of magnitude faster than the most effective and robust passive detector. Based on these findings, we recommend that, when both detector types are applicable, watermark-based detectors should be the preferred choice for their superior effectiveness, robustness, and efficiency. An interesting future work is to extend our ImageDetect-Bench to other modalities such as text, audio, and video.

## References

1. Rombach, R., Blattmann, A., Lorenz, D., Esser, P. & Ommer, B. *High-resolution Image Synthesis with Latent Diffusion Models* in *CVPR* (2022).
2. Podell, D. *et al.* *SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis* in *ICLR* (2024).
3. Saharia, C. *et al.* *Photorealistic text-to-image diffusion models with deep language understanding* in *NeurIPS* (2022).
4. Li, Z. *et al.* Hunyuan-DiT: A Powerful Multi-Resolution Diffusion Transformer with Fine-Grained Chinese Understanding. *arXiv preprint arXiv:2405.08748* (2024).
5. Clare, D. *Elon Musk’s AI Photo Tool is Generating Realistic, Fake Images of Trump, Harris and Biden* <https://www.cnn.com/2024/08/15/tech/elon-musk-x-grok-ai-images/index.html>. 2024.
6. Isaac, S.-B. & Naomi, N. *Fake Images of Trump Arrest Show ‘Giant Step’ for AI’s Disruptive Power* <https://www.washingtonpost.com/politics/2023/03/22/trump-arrest-deepfakes/>. 2023.
7. Joslin, M., Wang, X. & Hao, S. *Double Face: Leveraging User Intelligence to Characterize and Recognize AI-synthesized Faces* in *USENIX Security* (2024).
8. Sha, Z., Li, Z., Yu, N. & Zhang, Y. *De-fake: Detection and Attribution of Fake Images Generated by Text-to-image Generation Models* in *ACM CCS* (2023).
9. Ojha, U., Li, Y. & Lee, Y. J. *Towards Universal Fake Image Detectors That Generalize Across Generative Models* in *CVPR* (2023).
10. Ricker, J., Damm, S., Holz, T. & Fischer, A. *Towards the Detection of Diffusion Model Deepfakes* in *VISIGRAPP* (2022).
11. Sha, Z., Tan, Y., Li, M., Backes, M. & Zhang, Y. *ZeroFake: Zero-Shot Detection of Fake Images Generated and Edited by Text-to-Image Generation Models* in *ACM CCS* (2024).

12. Alam, I., Muneer, M. S. & Woo, S. S. *UGAD: Universal Generative AI Detector utilizing Frequency Fingerprints* in *ACM CIKM* (2024).
13. HIVE-AI. *HIVE AI-Generated Content Detection Tools* <https://hivemoderation.com/ai-generated-content-detection>. 2024.
14. ILLUMINARTY-AI. *ILLUMINARTY AI Generated Content Detection* <https://illuminarty.ai/en/image/ai-generated-image-detection.html>. 2024.
15. Tancik, M., Mildenhall, B. & Ng, R. *StegaStamp: Invisible Hyperlinks in Physical Photographs* in *CVPR* (2020).
16. Jiang, Z., Guo, M., Hu, Y., Jia, J. & Gong, N. Z. *Certifiably Robust Image Watermark* in *ECCV* (2024).
17. Wen, Y., Kirchenbauer, J., Geiping, J. & Goldstein, T. *Tree-Rings Watermarks: Invisible Fingerprints for Diffusion Images* in *NeurIPS* (2024).
18. Fernandez, P., Couairon, G., Jégou, H., Douze, M. & Furon, T. *The Stable Signature: Rooting Watermarks in Latent Diffusion Models* in *ICCV* (2023).
19. Kim, C., Min, K., Patel, M., Cheng, S. & Yang, Y. *WOUAF: Weight Modulation for User Attribution and Fingerprinting in Text-to-Image Diffusion Models* in *CVPR* (2024).
20. Zhu, J., Kaplan, R., Johnson, J. & Fei-Fei, L. *HiDDeN: Hiding Data with Deep Networks* in *ECCV* (2018).
21. Yang, Z. et al. *Gaussian Shading: Provable Performance-Lossless Image Watermarking for Diffusion Models* in *CVPR* (2024).
22. Jiang, Z., Guo, M., Hu, Y. & Gong, N. Z. Watermark-based Detection and Attribution of AI-Generated Content. *arXiv preprint arXiv:2404.04254* (2024).
23. The-White-House. *Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence* <https://www.whitehouse.gov/briefing-room/presidential-actions/2023/10/30/executive-order-on-the-safe-secure-and-trustworthy-development-and-use-of-artificial-intelligence/>. 2023.
24. OpenAI. *C2PA in DALL-E 3* <https://help.openai.com/en/articles/8912793-c2pa-in-dall-e-3>. 2024.
25. Google-DeepMind. *Identifying AI-generated content with SynthID* <https://deepmind.google/technologies/synthid/>. 2024.
26. Archer, K. *Watermarks in Preview in Azure OpenAI Service* <https://techcommunity.microsoft.com/blog/azure-ai-services-blog/watermarks-in-preview-in-azure-openai-service/4253344>. 2024.
27. Ha, A. Y. J. et al. *Organic or Diffused: Can We Distinguish Human Art from AI-generated Images?* in *ACM CCS* (2024).
28. Saberi, M. et al. *Robustness of AI-Image Detectors: Fundamental Limits and Practical Attacks* in *ICLR* (2024).
29. Meng, X., Wang, L., Guo, S., Ju, L. & Zhao, Q. *AVA: Inconspicuous Attribute Variation-based Adversarial Attack Bypassing Deepfake Detection* in *IEEE S&P* (2024).
30. Abdullah, S. M. et al. *An Analysis of Recent Advances in Deepfake Image Detection in an Evolving Threat Landscape* in *IEEE S&P* (2024).
31. Jiang, Z., Zhang, J. & Gong, N. Z. *Evading Watermark based Detection of AI-Generated Content* in *ACM CCS* (2023).
32. Hu, Y., Jiang, Z., Guo, M. & Gong, N. A Transfer Attack to Image Watermarks. *arXiv preprint arXiv:2403.15365* (2024).
33. An, B. et al. *WAVES: Benchmarking the Robustness of Image Watermarks* in *ICLR Workshop* (2024).
34. Hu, Y., Jiang, Z., Guo, M. & Gong, N. Stable Signature is Unstable: Removing Image Watermark from Diffusion Models. *arXiv preprint arXiv:2405.07145* (2024).
35. Hendrycks, D. & Dietterich, T. *Benchmarking Neural Network Robustness to Common Corruptions and Perturbations* in *ICLR* (2018).
36. Boyat, A. K. & Joshi, B. K. A Review Paper: Noise Models in Digital Image Processing. *SIPIJ* (2015).
37. Wallace, G. K. The JPEG Still Picture Compression Standard. *IEEE TCE* (1992).
38. Jacobsen, E. & Lyons, R. An Update to the Sliding DFT. *IEEE SPM* (2004).
39. Gedraite, E. S. & Hadad, M. *Investigation on the Effect of A Gaussian Blur in Image Filtering and Segmentation* in *IEEE ELMAR* (2011).
40. Chen, J., Jordan, M. I. & Wainwright, M. J. *HopSkipJumpAttack: A Query-Efficient Decision-Based Attack* in *IEEE S&P* (2020).

41. Andriushchenko, M., Croce, F., Flammarion, N. & Hein, M. *Square Attack: A Query-efficient Black-box Adversarial Attack via Random Search* in *ECCV* (2020).
42. Madry, A., Makelov, A., Schmidt, L., Tsipras, D. & Vladu, A. *Towards Deep Learning Models Resistant to Adversarial Attacks* in *ICLR* (2018).
43. Li, J., Li, D., Xiong, C. & Hoi, S. *BLIP: Bootstrapping Language-image Pre-training for Unified Vision-language Understanding and Generation* in *ICML* (2022).
44. Dhariwal, P. & Nichol, A. *Diffusion Models Beat GANs on Image Synthesis* in *NeurIPS* (2021).
45. Zhu, X., Cohen, S., Schiller, S. & Milanfar, P. Estimating Spatially Varying Defocus Blur from A Single Image. *IEEE TIP* (2013).
46. Hong, H. & Hong, Y. *Certifiable black-box attack: Ensuring provably successful attack for adversarial examples* in *ACM CCS* (2024).
47. Brendel, W., Rauber, J. & Bethge, M. *Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models* in *ICLR* (2018).
48. Ilyas, A., Engstrom, L. & Madry, A. *Prior Convictions: Black-box Adversarial Attacks with Bandits and Priors*. *ICLR* (2019).
49. Ilyas, A., Engstrom, L., Athalye, A. & Lin, J. *Black-box Adversarial Attacks with Limited Queries and Information* in *ICML* (2018).
50. Goodfellow, I. J., Shlens, J. & Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572* (2014).
51. Mooney, C. Z. *Monte Carlo Simulation* (Sage, 1997).
52. Lin, T.-Y. *et al.* *Microsoft COCO: Common Objects in Context* in *ECCV* (2014).
53. Sharma, P., Ding, N., Goodman, S. & Soricut, R. *Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset for Automatic Image Captioning* in *ACL* (2018).
54. Plummer, B. A. *et al.* *Flickr30k Entities: Collecting Region-to-phrase Correspondences for Richer Image-to-sentence Models* in *ICCV* (2015).
55. Sidorov, O., Hu, R., Rohrbach, M. & Singh, A. *Textcaps: A Dataset for Image Captioning with Reading Comprehension* in *ECCV* (2020).
56. Radford, A. *et al.* *Learning Transferable Visual Models from Natural Language Supervision* in *ICML* (2021).
57. Jaderberg, M., Simonyan, K., Zisserman, A., *et al.* *Spatial Transformer Networks* in *NeurIPS* (2015).
58. Zhang, R., Isola, P., Efros, A. A., Shechtman, E. & Wang, O. *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric* in *CVPR* (2018).
59. Karen, S. & Andrew, Z. *Very Deep Convolutional Networks for Large-scale Image Recognition* in *ICLR* (2014).
60. Scalability, H. *How Twitter Handles 3,000 Images Per Second* <https://highscalability.com/how-twitter-handles-3000-images-per-second/>. 2016.
61. Cohen, J., Rosenfeld, E. & Kolter, Z. *Certified Adversarial Robustness via Randomized Smoothing* in *ICML* (2019).
62. NVIDIA. *AI-Generated Image Detection* <https://build.nvidia.com/hive/ai-generated-image-detection/modelcard>. 2024.
63. Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D. & Finn, C. *Detectgpt: Zero-shot Machine-generated Text Detection Using Probability Curvature* in *ICML* (2023).
64. Liu, Z., Yao, Z., Li, F. & Luo, B. *On the Detectability of ChatGPT Content: Benchmarking, Methodology, and Evaluation through the Lens of Academic Writing* in *ACM CCS* (2024).
65. Pu, J. *et al.* *Deepfake Text Detection: Limitations and Opportunities* in *IEEE S&P* (2023).
66. Abdelnabi, S. & Fritz, M. *Adversarial Watermarking Transformer: Towards Tracing Text Provenance with Data Hiding* in *IEEE S&P* (2021).
67. Kirchenbauer, J. *et al.* *A Watermark for Large Language Models* in *ICML* (2023).
68. Dathathri, S. *et al.* Scalable Watermarking for Identifying Large Language Model Outputs. *Nature* (2024).
69. Li, X. *et al.* *Safeear: Content Privacy-preserving Audio Deepfake Detection* in *ACM CCS* (2024).
70. Blue, L. *et al.* *Who are you (i really wanna know)? detecting audio {DeepFakes} through vocal tract reconstruction* in *USENIX Security* (2022).
71. San Roman, R. *et al.* *Proactive Detection of Voice Cloning with Localized Watermarking* in *ICML* (2024).
72. Liu, C. *et al.* *Detecting Voice Cloning Attacks via Timbre Watermarking* in *NDSS* (2024).
73. Liu, H., Guo, M., Jiang, Z., Wang, L. & Gong, N. Z. *AudioMarkBench: Benchmarking Robustness of Audio Watermarking* in *NeurIPS* (2024).

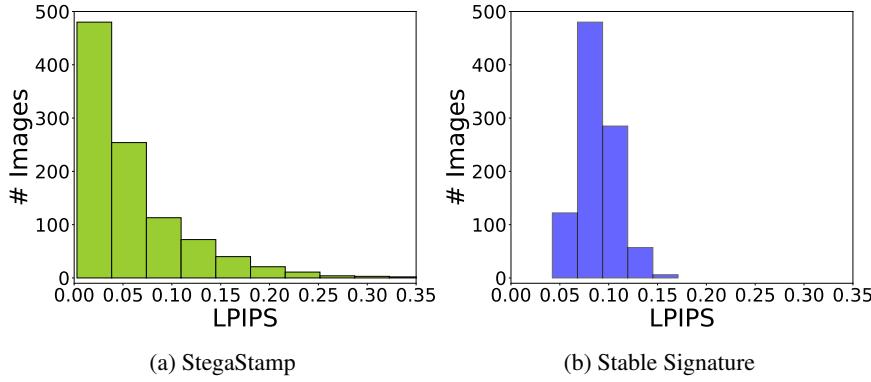


Figure 12: Distribution of LPIPS between AI-generated images and their watermarked counterparts for (a) StegaStamp and (b) Stable Signature on the MS dataset. Smoothed StegaStamp and StegaStamp use the same watermark encoder, and thus they have the same LPIPS distribution. The unwatermarked AI-generated images for Stable Signature are generated using the unmodified Stable Diffusion model with the same seeds. Since Tree-Ring modifies the seeds, there are no unwatermarked versions of watermarked AI-generated images and thus LPIPS is not applicable. The watermarks preserve visual quality well since the LPIPS values are small.

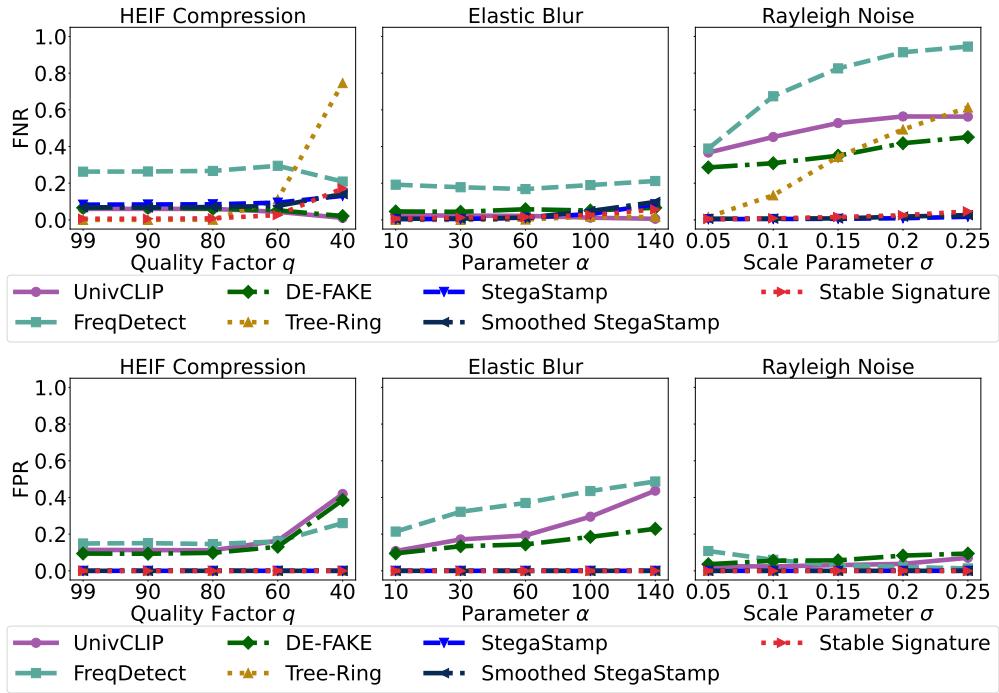


Figure 13: FNRs and FPRs of various detectors under 3 types of common perturbations unseen during training.

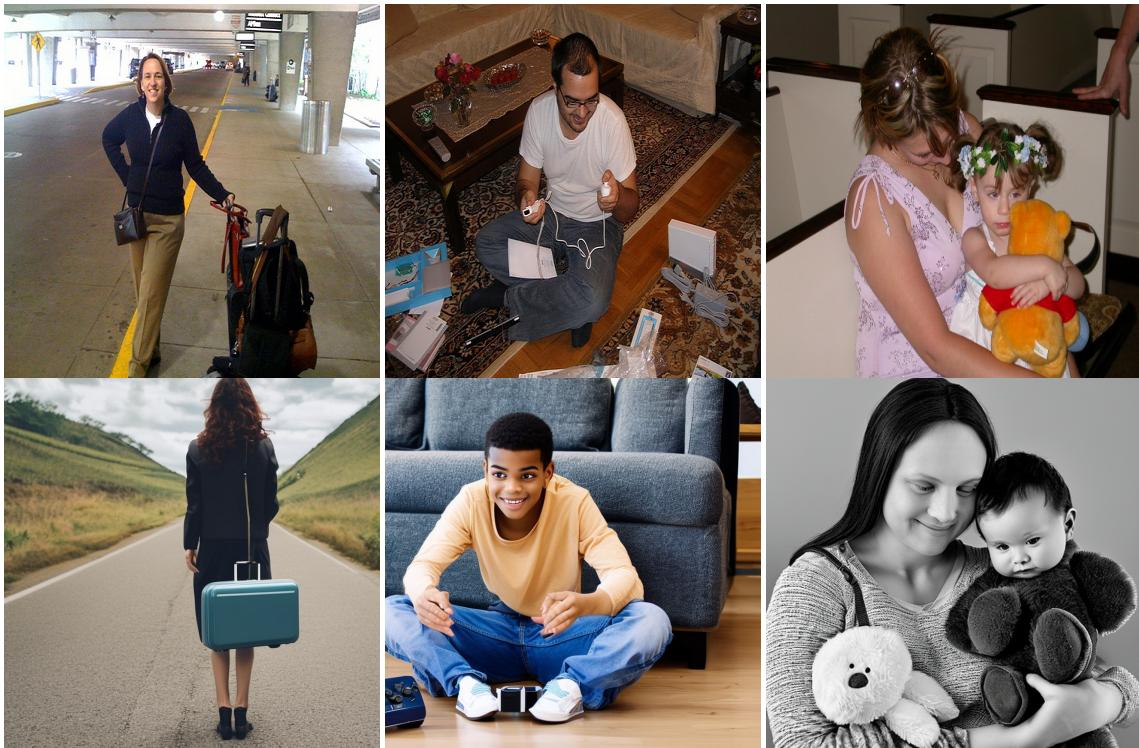


Figure 14: Examples of non-AI-generated (top row) and AI-generated (bottom row) images in the MS dataset. The captions of the non-AI-generated images are: **(a)** “a woman standing by the road with a suitcase.”, **(b)** “the young man sits on the floor to look at his new game system.”, and **(c)** “a mom and a baby who is holding a teddy bear.”. The AI-generated images are generated by Stable Diffusion using the captions as prompts.



Figure 15: Examples of non-AI-generated (top row) and AI-generated (bottom row) images in the GD dataset. The captions of the non-AI-generated images are **(a)** “a baby mallard duck was featured”, **(b)** “a backyard of boulders and trees”, and **(c)** “a basket of farm fresh eggs”. The AI-generated images are generated by DALL-E 3 using the captions as prompts.

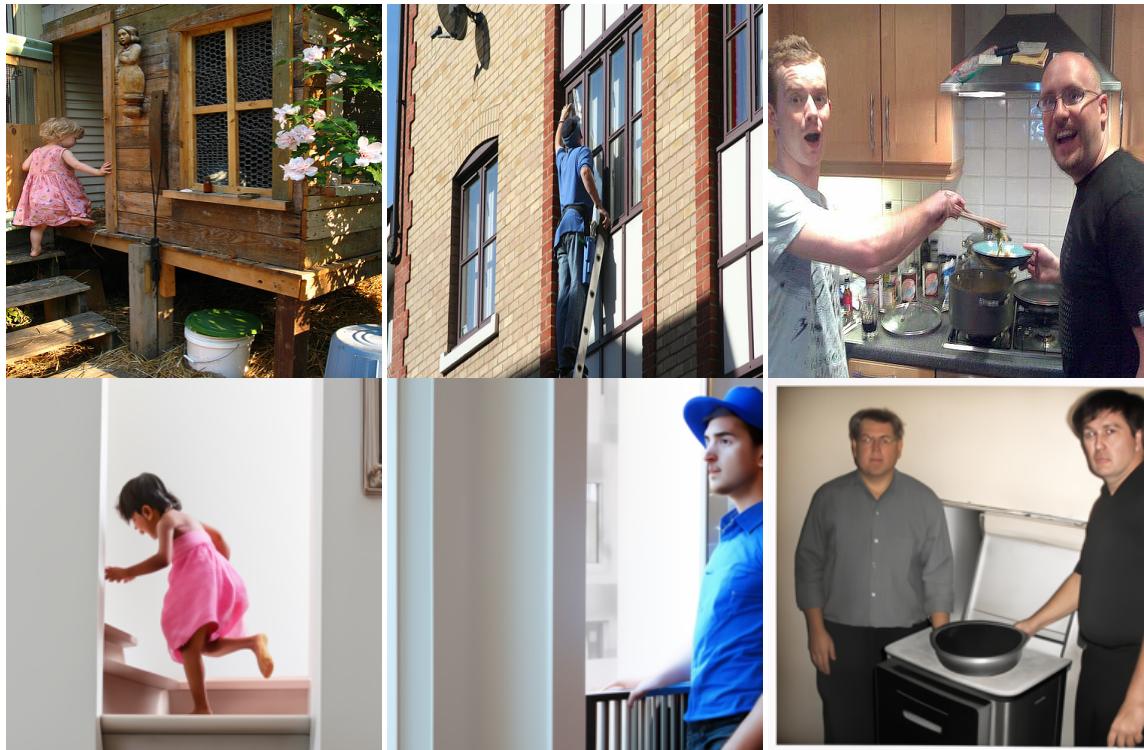


Figure 16: Examples of non-AI-generated (top row) and AI-generated (bottom row) images in the FD dataset. The captions of the non-AI-generated images are: **(a)** “A child in a pink dress is climbing up a set of stairs in an entry way.”, **(b)** “Someone in a blue shirt and hat is standing on stair and leaning against a window.”, and **(c)** “Two men, one in a gray shirt, one in a black shirt, standing near a stove.”. The AI-generated images are generated by DeepFloyd IF using the captions as prompts.



Figure 17: Examples of non-AI-generated (top row) and AI-generated (bottom row) images in the TH dataset. The captions of the non-AI-generated images are (a) “John C Fisher’s The Silver Slipper movie poster.”, (b) “A row of bottles of Poggio Antico wine on a store shelf.”, and (c) “The city of New York crowded with tourists with many ads on the buildings like H and M and Beautiful movie.”. The AI-generated images are generated by Hunyuan-DiT using the captions as prompts.

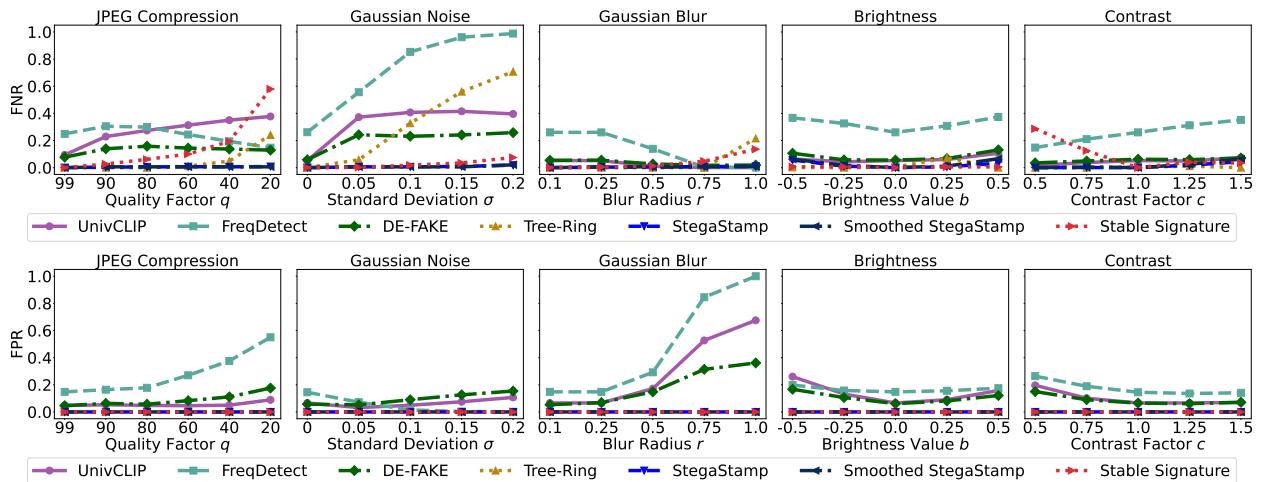


Figure 18: FNRs and FPRs of various detectors on the MS dataset under 5 types of common perturbations seen during training.

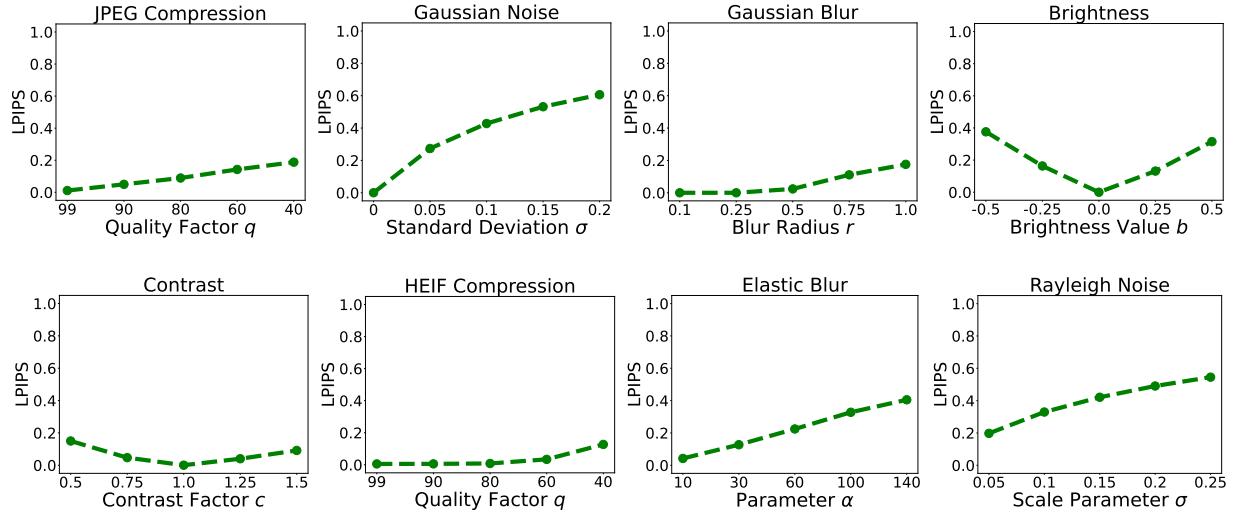


Figure 19: LPIPS under 8 types of common perturbations with varying parameter values.

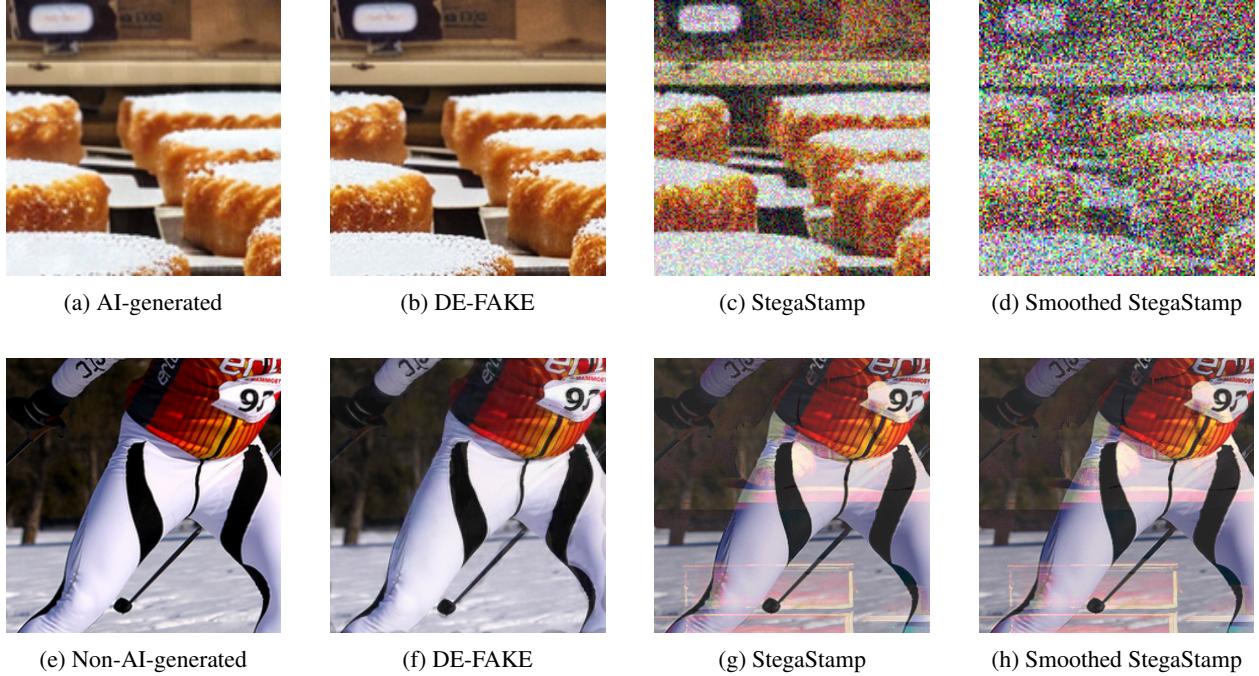


Figure 20: *First row:* an AI-generated image and its adversarial versions found by the HopSkipJump attack to make the best passive and watermark-based detectors misclassify it as non-AI-generated. The watermarked version of the AI-generated image is omitted since it visually looks the same. *Second row:* a non-AI-generated image and its adversarial versions found by the HopSkipJump attack to make the best passive and watermark-based detectors misclassify it as AI-generated. The HopSkipJump attack can deceive DE-FAKE in both removal and forgery attacks without sacrificing the image quality, while it substantially degrades the image quality when deceiving the watermark-based detectors especially Smoothed StegaStamp.