A. Technical Appendices and Supplementary Material

A.1. Details of Passive Detectors

We provide further details on the passive detectors introduced in Section 2.

A.1.1. Image-based Detectors

Image-based detectors [3, 36, 52, 58] rely solely on images as input and are typically categorized as *spatial-domain-based* or *frequency-domain-based*. Spatial-domain-based detectors [36, 52] extract features in the spatial domain for detection. In contrast, frequency-domain-based detectors [3, 58] 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 [52]: 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.

FreqDetect [58]: FreqDetect is a frequency-domain-based detector. This detector extracts frequency features using 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.

A.1.2. (Image, Caption)-based Detectors

These detectors [64, 65] 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 [64] as an example detector in this subcategory.

DE-FAKE [64]: 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 [40].

A.2. Details of Watermark-Based Detectors

We elaborate on the watermark-based detection methods introduced in Section 2.

A.2.1. Pre-generation Watermarks

Given a prompt and a seed, a typical text-to-image model iteratively denoises the seed as an image. Pre-generation watermarks [71, 72] 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 [71], in our experiments.

Tree-Ring [71]: 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 DDIM sampling [14]. 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 ℓ_1-norm distance between the decoded watermark and the ground-truth one is below the threshold τ, the image x is classified as watermarked and thus AI-generated.

A.2.2. Post-generation Watermarks

In post-generation watermarking methods [34, 68, 74], 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 pixel values, producing a watermarked image. In our experiments, we consider the state-of-the-art post-generation methods: StegaStamp [68] and its smoothed version [34].

StegaStamp [68]: 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 ℓ_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 τ, the image is classified as AI-generated.

Smoothed StegaStamp [34]: 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 decoder Dec extracts a watermark from each noisy image; and the ℓ_0-norm distance between each decoded watermark and the ground-truth watermark w_t is calculated. Finally, the image x is detected as AI-generated if the median of these N ℓ_0-norm distances is smaller than the threshold τ. Smoothed StegaStamp guarantees the detection result is unaffected by any ℓ_2-norm bounded perturbation added to x.

A.2.3. In-generation Watermarks

In-generation watermarks [15, 38] modify the parameters of a text-to-image model, ensuring that the generated images contain a watermark. In our experiments, we use Stable Signature[15], state-of-the-art in-generation watermark developed by Meta.

Stable Signature [15]: Stable Signature is based on the post-generation watermarking method HiDDeN [74]. 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 $\ell_0(w_t, Dec(x)) < \tau$.

A.3. Details of Perturbation Taxonomy

We provide additional details for the perturbations used in removal and forgery attacks, as discussed in Section 3.

A.3.1. Common Perturbations

These perturbations arise from common image processing operations. We categorize them into three types: lossy image compression [29, 70], pixel noise [8, 20], and visual effect [16, 75]. 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 [29, 70].

- **JPEG Compression [70].** 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) [29].** 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 σ, which controls the noise intensity.
- **Rayleigh noise.** Rayleigh noise is characterized by Rayleigh distribution with a scale parameter σ, 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 α, which defines the intensity of the transformation. Larger α causes more pronounced warping, while smaller α leads to subtler distortions.

A.4. 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 δ 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 [17] 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 δ until deceiving the detector or reaching the query budget. Based on the detector's response, these attacks can be further categorized into *decision-based* [9, 10, 22] and *score-based* [5, 26, 27]. 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 [10], a decision-based attack, and the Square attack [5], a score-based attack.

- **HopSkipJump (HSJ) attack [10].** Given an image x, this method begins with a large perturbation δ that successfully deceives the detector D, i.e., $D(x+\delta) \neq D(x)$. Such initial perturbation δ can be obtained via applying severe common perturbations onto the image x [32]. The perturbation δ 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 [50] to approximate its gradient. This gradient is

then used to search for the minimal perturbation capable of deceiving the detector.

- **Square attack [5].** 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 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 δ within the budget that is most likely to deceive the detector D. The process of finding this perturbation δ can be formulated as an optimization problem, typically solved using gradient-based methods [17, 47].

For passive detectors, the attacker optimizes δ such that the detector's prediction is opposite to the original prediction. The optimization problem is defined as follows [47]:

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

where CE denotes the cross-entropy loss of D for $x+\delta$ when treating $D(x)$ as its label, and $\|\cdot\|_\infty$ indicates ℓ_∞-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. [32], the attacker optimizes a perturbation δ 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:

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

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

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

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)* [47] to optimize δ. Specifically, we optimize δ under an ℓ_∞-norm constraint defined by the perturbation budget r. If δ exceeds this budget, it is projected back onto the ℓ_∞-norm ball.

A.5. Dataset Details

We provide detailed information for each of the four datasets used in our study, including sampling strategy, caption sources, generation models, and caption statistics.

MSCOCO – Stable Diffusion (MS): MSCOCO [43] 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 [59]. 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 [66] 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 [61]. 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 [67] 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 [42]. 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. It is worth noting that our training setup gives advantages to passive detectors.

A.6. Detector Training Details

A.6.1. Passive Detectors

We evaluate three passive detectors: UnivCLIP [52], FreqDetect [58], and DE-FAKE [64]. 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 [57] 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 3×10^{-4} and decays to 1×10^{-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 as the classifier. The logistic regression classifier is trained using the L-BFGS solver over 1,000 iterations with an ℓ_2-norm penalty. We also apply training with perturbations to enhance robustness.

DE-FAKE: DE-FAKE uses the ViT-B/32 CLIP model [57] to extract features from both images and captions, with a three-layer multilayer perceptron 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 by the Adam optimizer with a batch size of 500 for 800 iterations. The learning rate is initially set to 3×10^{-4} and decays to 1×10^{-6} by a cosine decay schedule. Additionally, training with perturbations is used to enhance robustness. For testing, we use BLIP [40] to generate a caption for each image and classify the resulting image-caption pair with the classifier.

A.6.2. Watermark-based Detectors

In our experiments, we use four watermark-based detectors: Tree-Ring [71], StegaStamp [68], Smoothed StegaStamp [34], and Stable Signature [15]. 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. [71]. 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. [71] 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 [43]. 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. [68], we employ a U-Net architecture as the watermark encoder Enc and a spatial transformer network [30] 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 [34] 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 ℓ_0-norm distance with the ground-truth watermark w_t. We then take the median across the 100 ℓ_0-norm distances as the final ℓ_0-norm distance, and classify the image x as AI-generated if the final ℓ_0-norm

distance is smaller than a threshold τ.

Stable Signature: For Stable Signature, we use the publicly available text-to-image model [15]. 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 [74]. 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 τ: Passive detectors rely on standard classifiers with widely used detection thresholds. In contrast, via properly setting the threshold τ, 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 [32], we select τ in our experiments to ensure this probability is less than 0.0001. Specifically, for StegaStamp and Smoothed StegaStamp, τ is set to 0.8125; for Stable Signature, τ is set to 0.7708; and for Tree-Ring, τ 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.

A.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 [46, 49, 56] and watermark-based detectors [1, 13, 31, 35, 39] have been proposed. For AI-generated audio, various passive [7, 41] and watermark-based detectors [44, 45, 62] 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 [47]. 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 [32], i.e., a larger adversarial perturbation is needed to deceive a detector, it is still insufficient, i.e., the adversarial perturbation is still small.

A.8. Broader Impacts

Our work addresses the critical ethical challenges posed by AI-generated images, particularly their misuse for spreading misinformation, impersonation, and manipulating public opinion. The risks of such misuse are evident, as AI-generated images have already been used to fabricate false events involving public figures. To mitigate these risks, our research focuses on benchmarking both passive and watermark-based AI-generated image detectors. This effort aims to enhance the reliability and robustness of detection while also safeguarding copyright.

Table 4. 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	$1.39 \cdot 10^{-2}$

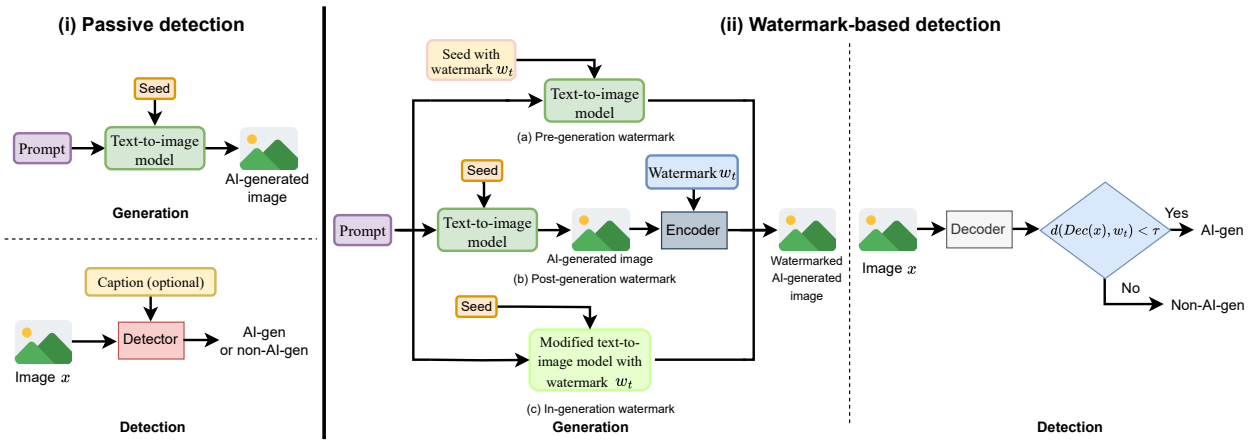


Figure 8. Illustration of passive and watermark-based detection of AI-generated images.

Figure 9. 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 10. 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.

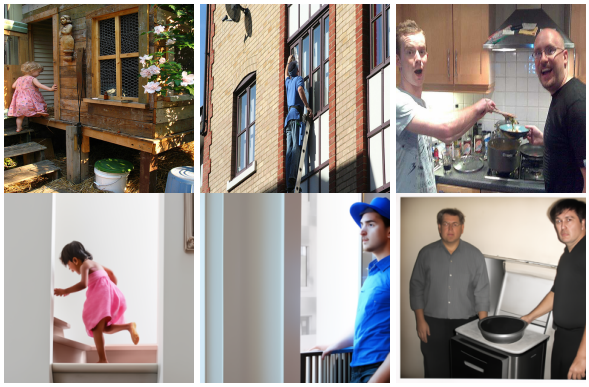
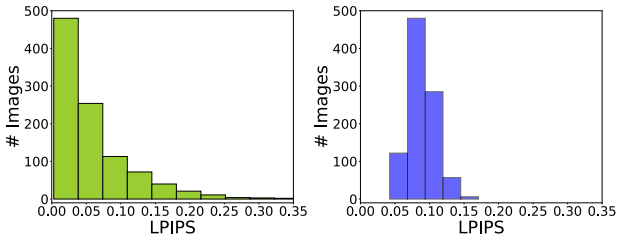


Figure 11. Examples of non-AI-generated (top row) and AI-generated (bottom row) images in the FD dataset. The captions of the non-AI-generated images are: **(a)** "A child in a pink dress is climbing up a set of stairs in an entry way.", **(b)** "Someone in a blue shirt and hat is standing on stair and 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 12. 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.



(a) StegaStamp (b) Stable Signature

Figure 13. 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.

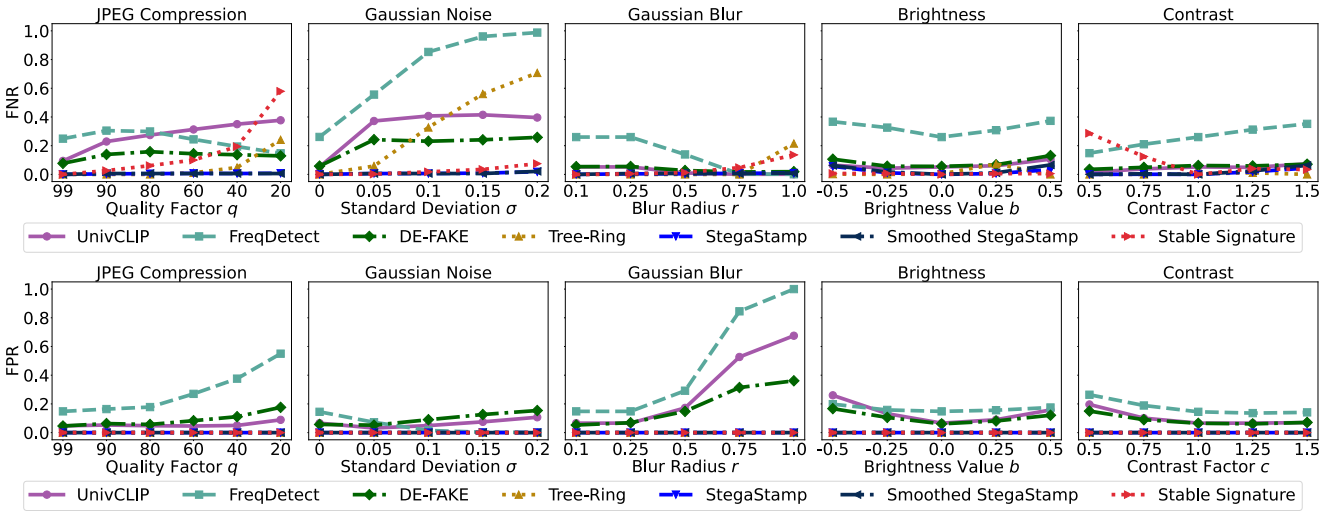


Figure 14. FNRs and FPRs of various detectors on the MS dataset under 5 types of common perturbations seen during training.

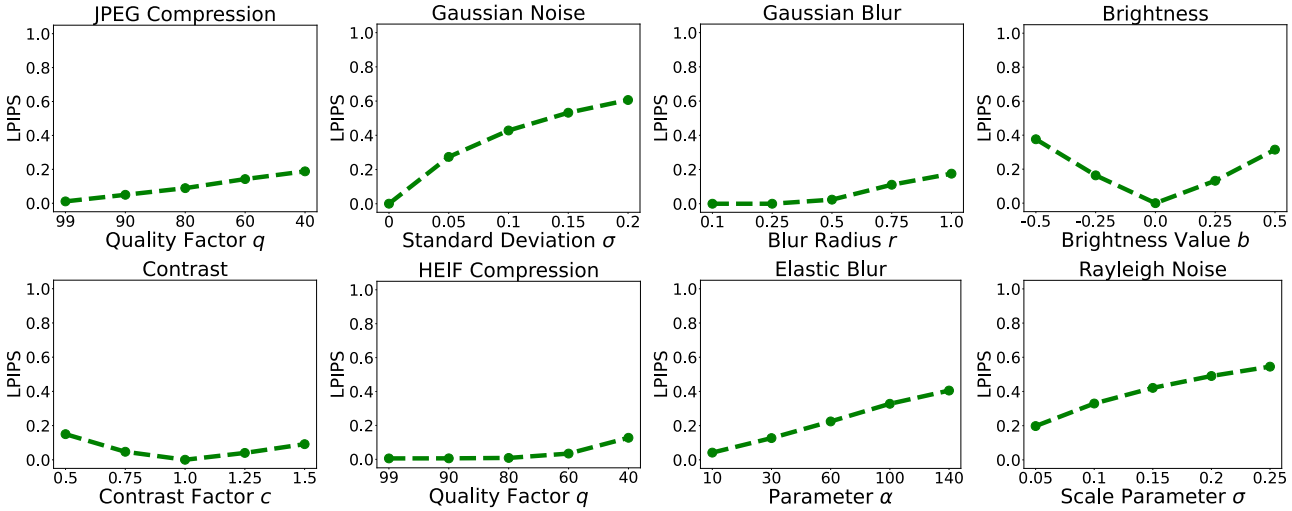


Figure 15. LPIPS under 8 types of common perturbations with varying parameter values.

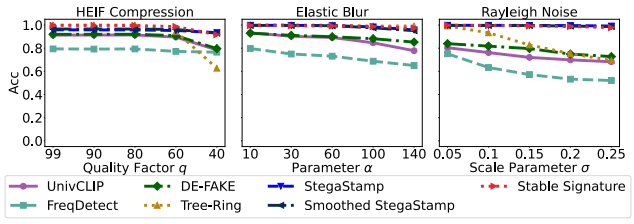


Figure 16. ACCs of various detectors under 3 types of common perturbations unseen during training.

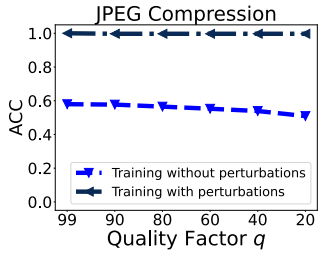
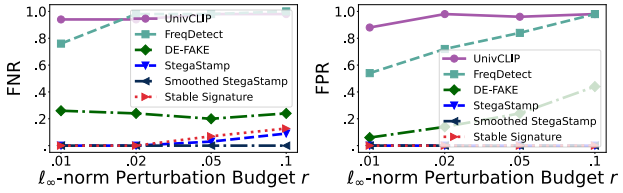
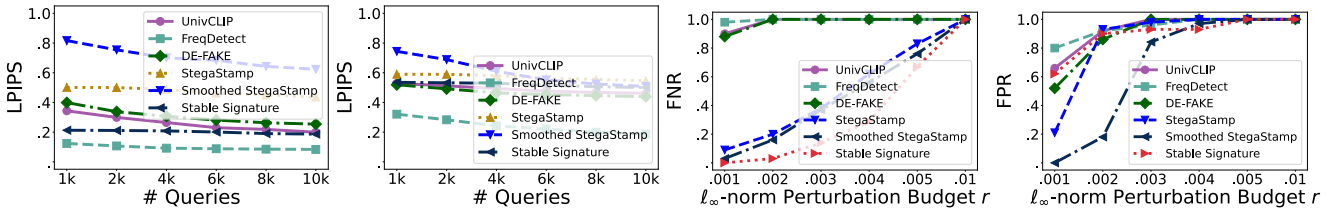


Figure 17. ACC of Smoothed StegaStamp on the MS dataset under JPEG compression with different quality factors when training with vs. without perturbations.



(a) Removal attack (b) Forgery attack

Figure 18. FNRs and FPRs of various detectors under removal and forgery attacks involving black-box Square attack with varying ℓ_∞-norm perturbation budgets r.



(a) Removal attack (b) Forgery attack (c) Removal attack (d) Forgery attack

Figure 19. LPIPS for various detectors under (a) removal and (b) forgery attacks involving the black-box HopSkipJump attack with varying query budget. FNRs and FPRs of various detectors under (c) removal and (d) forgery attacks involving white-box adversarial perturbations with varying ℓ_∞-norm perturbation budget r.

(a) AI-generated (b) DE-FAKE (c) StegaStamp (d) Smoothed StegaStamp

(e) Non-AI-generated (f) DE-FAKE (g) StegaStamp (h) Smoothed StegaStamp

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.