Evading Watermark based Detection of AI-Generated Content

Zhengyuan Jiang*
Duke University
zhengyuan.jiang@duke.edu

Jinghuai Zhang*
Duke University
jinghuai.zhang@duke.edu

Neil Zhenqiang Gong
Duke University
neil.gong@duke.edu

ABSTRACT

A generative AI model can generate extremely realistic-looking content, posing growing challenges to the authenticity of information. To address the challenges, watermark has been leveraged to detect AI-generated content. Specifically, a watermark is embedded into an AI-generated content before it is released. A content is detected as AI-generated if a similar watermark can be decoded from it. In this work, we perform a systematic study on the robustness of such watermark-based AI-generated content detection. Our work shows that an attacker can post-process a watermarked image via adding a small, human-imperceptible perturbation to it, such that the post-processed image evades detection while maintaining its visual quality. We show the effectiveness of our attack both theoretically and empirically. Moreover, to evade detection, our adversarial post-processing method adds much smaller perturbations to AI-generated images and thus better maintain their visual quality than existing popular post-processing methods such as JPEG compression, Gaussian blur, and Brightness/Contrast. Our work shows the insufficiency of existing watermark-based detection of AI-generated content, highlighting the urgent needs of new methods. Our code is publicly available: https://github.com/zhengyuan-jiang/WEvade.

CCS CONCEPTS

• **Security and privacy** → **Security services**; • **Social and professional topics** → *Intellectual property*.

KEYWORDS

AI-generated content detection; Watermarking; Robustness.

ACM Reference Format:
Zhengyuan Jiang*, Jinghuai Zhang*, and Neil Zhenqiang Gong. 2023. Evading Watermark based Detection of AI-Generated Content. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS'23).* ACM, New York, NY, USA, 19 pages. https://doi.org/10.475/XXXXXXXX

1 INTRODUCTION

Given a prompt, generative AI–such as DALL-E, Stable Diffusion, and ChatGPT–can generate extremely realistic looking content including image and text. Like any advanced technology, generative AI is also a double-edged sword. On one hand, generative AI can assist human to enhance effectiveness and efficiency in various domains such as searching, art image creation, and character design in online games. The market for generative AI was predicted to increase to 50 billion by 2028 [19]. On the other hand, generative AI also raises many ethical concerns. For instance, their generated realistic looking content can be used to aid disinformation campaigns on social media; they are disruptive for learning and education as students can use them to complete/aid homework and exams; and people can use them to generate content and claim its ownership/copyright, though not allowed by US Copyright Office [2].

Watermark-based detection [1, 9, 14, 39, 41] of AI-generated content is a key technology to address these ethical concerns. Multiple AI companies–such as OpenAI, Google, and Meta–have made voluntary commitments to watermark AI-generated content [18]. In particular, a watermark is embedded into an AI-generated content when it is generated. The watermark enables proactive detection of AI-generated content in the future: a content is AI-generated if a similar watermark can be extracted from it. In this work, we focus on AI-generated images. For instance, DALL-E embeds a *visible* watermark at the bottom right corner of its generated images (Figure 29 in Appendix shows an example); Stable Diffusion uses a *non-learning-based* watermarking method [24] to embed an invisible watermark into generated images; and Meta [9] proposed to use *learning-based* watermarking methods.

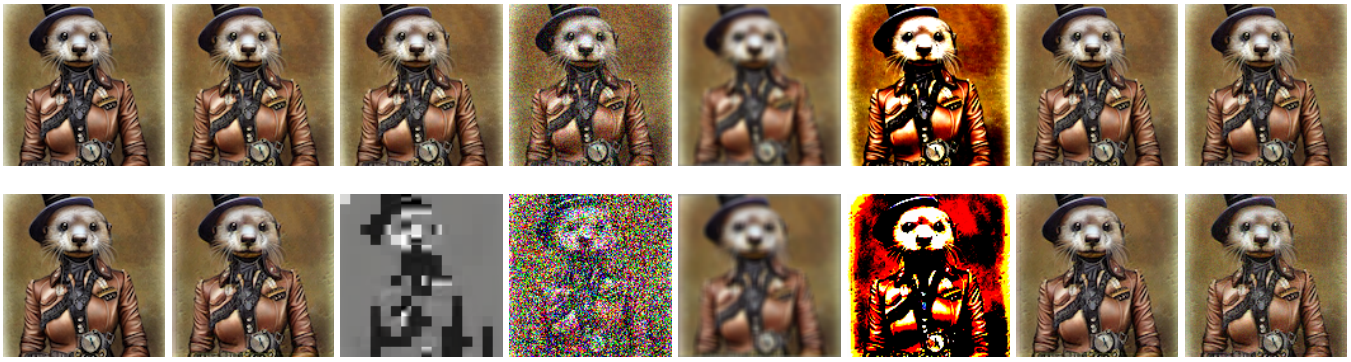
A watermarking method [3, 16, 23, 24, 34, 37, 42, 44, 46] consists of three key components, i.e., *watermark* (we represent it as a bitstring), *encoder*, and *decoder*. Given an image and a watermark, an encoder embeds the watermark into the image to produce a *watermarked image*; and a decoder decodes a watermark from an image (a watermarked image or an *original image* without watermark). We note that some watermarking methods [9, 38, 41] embed the encoder into a generative AI model, so the watermark is already embedded into its generated images at generation. An image is predicted as AI-generated if the *bitwise accuracy* of the decoded watermark is larger than a threshold τ, where bitwise accuracy is the fraction of matched bits in the decoded watermark and the ground-truth one. The threshold τ should be larger than 0.5 since the bitwise accuracy of original images without watermarks would be around 0.5. In a non-learning-based watermarking method [3, 23, 24], which has been studied for decades, both encoder and decoder are designed based on heuristics, while they are neural networks and automatically learnt using a set of images in learning-based watermarking methods [16, 34, 37, 42, 44, 46], an emerging category of watermarking methods.

Robustness against *post-processing*, which post-processes an AI-generated image, is crucial for a watermark-based detector. Unfortunately, the visible watermark adopted by DALL-E can be easily removed without sacrificing the image quality at all [20]. Non-learning-based watermarks (e.g., the one used by Stable Diffusion) can be removed by popular image post-processing methods (e.g.,

*Equal contribution.

CCS'23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 123-4567-24-567/08/09…$15.00
https://doi.org/10.475/XXXXXXXX



(a) Original (b) Watermarked (c) JPEG (d) GN (e) GB (f) B/C (g) WEvade-W-II (h) WEvade-B-Q

Figure 1: Illustration of original image, watermarked image, and watermarked images post-processed by existing and our methods (last two columns) to evade detection. The watermarking method is HiDDeN. GN: Gaussian noise. GB: Gaussian blur. B/C: Brightness/Contrast. The encoder/decoder are trained via standard training (*first row*) or adversarial training (*second row*).

JPEG compression) [9, 46], which we also confirm in our experiments in Section 7.5. Learning-based watermarking methods were believed to be robust against post-processing [9, 16, 42, 46]. In particular, the encoder and decoder can be trained using *adversarial training* [12] to enhance robustness against post-processing. In adversarial training, a post-processing layer is added between the encoder and decoder; it post-processes a watermarked image outputted by the encoder before feeding it into the decoder; and the encoder and decoder are adversarially trained such that the watermark decoded from a post-processed watermarked image is still similar to the ground-truth one. However, existing studies only evaluated the robustness of learning-based watermarking methods against popular image post-processing methods such as JPEG compression, Gaussian blur, and Brightness/Contrast, leaving their robustness against adversarial post-processing unexplored.

Our work: We aim to bridge this gap in this work. We propose WEvade, an adversarial post-processing method to evade watermark-based detection of AI-generated images. WEvade adds a small, human-imperceptible perturbation to a watermarked image such that the perturbed image is falsely detected as non-AI-generated. WEvade can be viewed as adversarial examples [33] to watermarking methods. However, as we discuss below, simply extending standard adversarial examples to watermarking is insufficient. WEvade considers the unique characteristics of watermarking to construct adversarial examples.

White-box setting. In this threat model, we assume the attacker has access to the decoder used by detectors, but no access to the ground-truth watermark and encoder. Given a watermarked image generated by an AI model, an attacker aims to post-process it via adding a small perturbation to it, such that detectors with any threshold $\tau > 0.5$ would falsely detect the post-processed watermarked image as non-AI-generated. One way (denoted as WEvade-W-I) to achieve the goal is to simply extend the standard adversarial examples to the decoder. In particular, an attacker finds the perturbation such that each bit of the decoded watermark flips, leading to a very small bitwise accuracy and thus evasion. However, we show that such attack can be mitigated by a *double-tail detector*, which we propose to detect an image as AI-generated if the decoded watermark has either too small or too large bitwise accuracy.

To address the challenge, we propose WEvade-W-II, which adds perturbation to a watermarked image such that the decoded watermark has a bitwise accuracy close to 0.5, making the post-processed image indistinguishable with original images without watermarks. However, since the attacker does not know the ground-truth watermark, it is challenging to measure the bitwise accuracy of the decoded watermark. Our key observation to address the challenge is that a watermark selected uniformly at random would have a bitwise accuracy close to 0.5, no matter what the ground-truth watermark is. Based on this observation, we find the perturbation with which the decoded watermark is close to a random watermark. We formulate finding such perturbation as an optimization problem and propose a solution to solve it.

Black-box setting. In this threat model, we assume the attacker can only query the detector API, which returns a binary result ("AI-generated" or "non-AI-generated") for any image. One way (called WEvade-B-S) to evade detection is that the attacker trains a surrogate encoder and decoder using a watermarking algorithm. Then, given a watermarked image, the attacker finds the perturbation based on the surrogate decoder using the white-box attack WEvade-W-II. However, such attack achieves limited evasion rates because the surrogate decoder and target decoder output dissimilar watermarks for an image.

To address the challenge, we propose WEvade-B-Q, which extends state-of-the-art hard-label based adversarial example technique called HopSkipJump [6] to watermark-based detector. Given a watermarked image, HopSkipJump can iteratively find a post-processed version to evade detection via just querying the detector API. Specifically, starting from a *random* initial image that is predicted as non-AI-generated by the detector, HopSkipJump iteratively moves the image closer to the given watermarked image to reduce the added perturbation while always guaranteeing that the image evades detection. Essentially, in each iteration, HopSkipJump returns 1) a perturbation to update the image and 2) the number of queries to the detector API used to find such perturbation. The iterative process stops when HopSkipJump uses a given *query budget*. However, simply applying HopSkipJump to watermarking may end up with a large perturbation. The reasons include 1) the random initial image may be far away from the given watermarked image,

and 2) the iterative process does not always reduce the perturbation, and thus an improper setting of query budget may actually enlarge the perturbation. To address the challenges, our WEvade-B-Q constructs the initial image using the watermarked image post-processed by popular methods such as JPEG compression, which results in an initial image closer to the watermarked image. Moreover, WEvade-B-Q stops the iterative process when the added perturbation starts to increase, which reduces both perturbation and number of queries to the detector API.

Theoretical and empirical evaluation. Theoretically, we derive the evasion rates of different variants of WEvade. For instance, WEvade-W-I achieves evasion rate of 1 against the standard *single-tail detector*, but its evasion rate reduces to 0 when our proposed double-tail detector is used. We also derive a lower bound of the evasion rate of WEvade-W-II using triangle inequality. Moreover, we derive the evasion rate of WEvade-B-S based on a formal similarity quantification between the watermarks outputted by the surrogate decoder and target decoder. We also show that WEvade-B-Q achieves evasion rate of 1.

Empirically, we evaluate our attacks using multiple datasets and multiple watermarking methods, including two learning-based ones (HiDDeN [46] and UDH [42]) and the non-learning-based one adopted by Stable Diffusion [24]. Our results show that our method is effective and outperforms existing post-processing methods. In particular, existing post-processing methods need to add much larger perturbations in order to achieve evasion rates comparable to our method. We find that adversarial training can enhance robustness of watermarking, i.e., a post-processing method needs to add larger perturbation to evade detection. However, the perturbation added by our method is still small and maintains image quality, indicating the insufficiency of adversarial training. Figure 1 shows an original image, its watermarked version, and the watermarked versions post-processed by different methods such that the decoded watermarks achieve bitwise accuracy close to 0.55 (indistinguishable with original images without watermarks). The results show that existing post-processing methods substantially sacrifice image quality to evade a watermark-based detector based on adversarial training, while our methods still maintain image quality.

To summarize, our key contributions are as follows:

- We propose WEvade , which adds small, human-imperceptible perturbations to AI-generated images to evade watermark-based detectors.
- We theoretically analyze the evasion rates of WEvade in both white-box and black-box settings.
- We empirically evaluate WEvade on multiple watermarking methods and datasets in various scenarios.

2 RELATED WORK

2.1 Detecting AI-generated Content

Generative AI models could be GANs [11], diffusion models (e.g., DALL-E [25], Stable Diffusion [27]), or language models (e.g., ChatGPT [22]). AI-generated content could be image (our focus in this work) or text. AI-generated content detection include *passive detection* [10, 21, 29, 35, 40, 45] and *proactive detection* [3, 16, 23, 24, 34, 37, 42, 44, 46]. Passive detection aims to leverage statistical artifacts in AI-generated content to distinguish them with non-AI-generated content, while proactive detection aims to proactively embed a watermark into AI-generated content when it is generated, which enables detection in the future. Several studies [4, 7, 28] showed that passive detectors are not robust to evasion attacks, i.e., an attacker can slightly perturb an AI-generated content to remove the statistical artifacts exploited by a passive detector and thus evade detection. However, the robustness of proactive detectors against evasion attacks is much less explored. For instance, recent studies [9] suggested that proactive detectors are more robust than passive ones. Our work focuses on proactive detectors and shows that they are not as robust as previously thought.

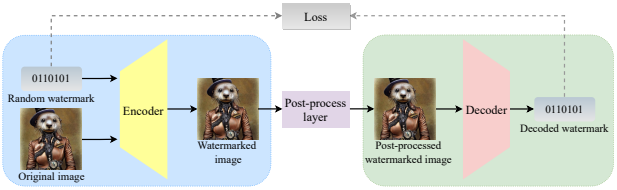


Figure 2: Illustration of training encoder and decoder in learning-based watermarking methods.

2.2 Watermarking Methods

Since we focus on AI-generated images, we review image watermarking methods. A watermarking method has three key components: *watermark*, *encoder*, and *decoder*. We consider a watermark w as a n-bit bitstring, e.g., $w = 0110101$. An encoder takes an image I and a watermark w as input and produces a watermarked image I_w. Formally, we have $I_w = E(I, w)$, where E stands for encoder. A decoder takes an image as input and outputs a watermark. Formally, we have $w_I = D(I)$. Note that, given any image (e.g., an original image without watermark or a watermarked image) as input, the decoder can output a watermark. Watermarking methods can be categorized into two groups depending on how the encoder and decoder are designed, i.e., *non-learning-based* and *learning-based*.

Non-learning-based methods: In these methods [3, 23, 24], the encoder and decoder are hand-crafted based on heuristics. Non-learning-based methods have been studied for around three decades. *Invisible-watermark* [24] is a representative non-learning-based method, which is adopted by Stable Diffusion. Roughly speaking, this method uses Discrete Wavelet Transform (DWT) to decompose an image into several frequency sub-bands, applies Discrete Cosine Transform (DCT) to each block of some carefully selected sub-bands, and alters certain frequency coefficients of each block via adding a bit of the watermark. The watermark is embedded in selected frequency sub-bands of the image, and the watermarked image is obtained via inverse transform.

Learning-based methods: In these methods [16, 34, 37, 42, 44, 46], the encoder and decoder are neural networks and automatically learnt via deep learning techniques. Roughly speaking, the second-to-last layer of the decoder outputs a vector of real-value numbers, each entry of which indicates the likelihood that the corresponding bit of the watermark is 1. Formally, we denote by $F(I)$ such vector for an image I, where $F(I)_i$ is the likelihood that the ith bit of the decoded watermark is 1; and the decoded watermark w_I is obtained by thresholding $F(I)$, i.e., the ith bit of w_I is 1 if and only if $F(I)_i > 0.5$. HiDDeN [46] and UDH [42] are two representative learning-based methods. In HiDDeN, the encoder concatenates a

watermark and an image to produce a watermarked image. In UDH, the encoder transforms the watermark into a QR code, maps the QR code to a secret image which has the same size as an original image, and pixel-wisely adds the secret image to an original image as a watermarked image. Figure 2 illustrates how the encoder and decoder are trained, which we discuss next.

Standard training. The encoder and decoder are iteratively trained using a set of images and the standard Stochastic Gradient Descent (SGD) algorithm. In each iteration, a mini-batch of images are used to update the encoder and decoder. Specifically, for each image I in the mini-batch, a random watermark w_I is sampled. The encoder E produces a watermarked image $E(I, w_I)$ for each image I and the corresponding random watermark w_I. The decoder D takes each watermarked image $E(I, w_I)$ as input and outputs a watermark $D(E(I, w_I))$. The encoder and decoder are learnt such that the decoded watermark $D(E(I, w_I))$ is close to w_I. In particular, they are updated via SGD to minimize a loss function $\sum_I loss(D(E(I, w_I)), w_I)$.

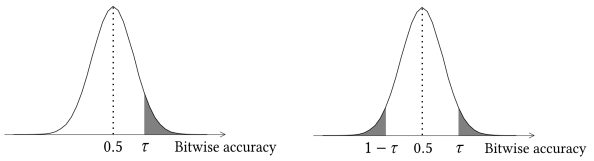
Adversarial training. A key advantage of learning-based methods is that they can leverage adversarial training [12, 17] to enhance their robustness against post-processing [16, 37]. Specifically, as illustrated in Figure 2, a post-processing layer is added between the encoder and decoder, which post-processes each watermarked image before feeding it to the decoder during training. For each image in a mini-batch during training, a post-processing method is randomly selected from a given set of ones, e.g., JPEG compression, Gaussian noise, Gaussian blur, Brightness/Contrast, and our WEvade. The encoder and decoder are updated via SGD to minimize a loss function $\sum_I loss(D(E(I, w_I)+\delta_I), w_I)$, where δ_I is the perturbation introduced by the post-processing method to the watermarked image $E(I, w_I)$. As shown by previous works [16, 37] and confirmed by our experiments, adversarial training makes learning-based watermarking robust against popular post-processing methods. However, it is still vulnerable to our adversarial post-processing method.

We note that some watermarking methods [9, 38, 41] embed the encoder into a generative AI model, so its generated images are already embedded with the watermark, but they still rely on the decoder for detection. For instance, Fernandez et al. [9] trains encoder/decoder using HiDDeN, embeds the encoder into image generator via fine-tuning it, and uses the decoder for detection. Our attacks are also applicable to such watermarking methods since they are agnostic to how a watermark is embedded into an AI-generated image.

3 WATERMARK-BASED DETECTORS

We formally define the detection setup and the standard *single-tail detector*. Moreover, we propose a *double-tail detector*, which can defend against the evasion attack (discussed in Section 5.1) that simply extends standard adversarial examples to watermarking.

Detection setup: We use I to denote an *image*, I_o to denote an *original image* without watermark, I_w to denote a *watermarked image*, and I_{pw} to denote a *post-processed watermarked image*. Note that, in our notations, I could be an I_o, I_w, or I_{pw}. We use $BA(w_1, w_2)$ to denote the *bitwise accuracy* of watermark w_1 compared to watermark w_2, i.e., $BA(w_1, w_2)$ is the fraction of bits that match in w_1 and w_2. Suppose a service provider (e.g., OpenAI) deploys a



(a) Single-tail detector (b) Double-tail detector

Figure 3: Illustration of (a) single-tail detector and (b) double-tail detector with threshold τ. The bitwise accuracy of an original image I_o follows a binomial distribution divided by n, i.e., $BA(D(I_o), w) \sim B(n, 0.5)/n$. The area of the shaded region(s) is the *false positive rate (FPR)* of a detector.

generative AI model (e.g., a text-to-image generative model) as a cloud service and has a ground-truth watermark w. Given a user query (known as *prompt*), the cloud service uses the AI model to generate an image, embeds its watermark w into it using the encoder (or the generated image already has watermark w [9, 38, 41]), and returns the watermarked image to the user. In such cloud service, detecting AI-generated images reduces to detecting watermarked images. Specifically, given an image I, we can decode a watermark $D(I)$ using the decoder. Then, we calculate the bitwise accuracy $BA(D(I), w)$ of the watermark $D(I)$ with respect to the ground-truth watermark w. A watermark-based detector (shown in Figure 3) leverages the bitwise accuracy to detect watermarked images, which we discuss below.

Single-tail detector: In the standard single-tail detector [9, 41], an image I is predicted as AI-generated if the bitwise accuracy of its decoded watermark is larger than a threshold τ, i.e., $BA(D(I), w) > \tau$, where w is the ground-truth watermark. A key challenge is how to set the threshold τ such that the *false positive rate (FPR)*, i.e., the probability that an original image is falsely detected as AI-generated, is bounded by a small value η, e.g., $\eta = 10^{-4}$. This challenge can be addressed by formally analyzing the relationship between the threshold τ and the FPR of the single-tail detector [9, 41].

Suppose $BA(D(I_o), w) = \frac{m}{n}$ for an original image I_o, where n is the length (i.e., number of bits) of the watermark and m is the number of matched bits between $D(I_o)$ and w. The key idea is that the service provider should pick the ground-truth watermark w uniformly at random. Thus, the decoded watermark $D(I_o)$ is not related to the randomly picked w, and each bit of $D(I_o)$ matches with the corresponding bit of w with probability 0.5. As a result, m is a random variable and follows a *binomial distribution* $B(n, 0.5)$. Therefore, the FPR (denoted as $FPR_s(\tau)$) of the single-tail detector with threshold τ can be calculated as follows [9, 41]:

$$\begin{aligned} FPR_s(\tau) &= \Pr(BA(D(I_o), w) > \tau) \\ &= \Pr(m > n\tau) = \sum_{k=\lceil n\tau \rceil}^{n} \binom{n}{k} \frac{1}{2^n}, \end{aligned} \tag{1}$$

where $FPR_s(\tau)$ is defined for any original image and the randomness in calculating the probability stems from picking the ground-truth watermark w uniformly at random. Thus, to make $FPR_s(\tau) < \eta$, τ should be at least $\tau^* = \arg\min_\tau \sum_{k=\lceil n\tau \rceil}^{n} \binom{n}{k} \frac{1}{2^n} < \eta$. For instance, when $n = 256$ and $\eta = 10^{-4}$, we have $\tau \geq \tau^* \approx 0.613$.

Double-tail detector: The single-tail detector can be easily evaded by simply extending standard adversarial examples to watermarking. In particular, a standard adversarial example based evasion attack adds perturbation to a watermarked image such that the decoded watermark has a very small bitwise accuracy, e.g., close to 0. However, we propose a double-tail detector to detect such perturbed images. Our key observation is that the watermarks decoded from original images have bitwise accuracy close to 0.5, while those decoded from watermarked images have large bitwise accuracy, e.g., close to 1. Thus, if the bitwise accuracy of the watermark decoded from an image is significantly smaller than 0.5, it is likely to be an adversarially perturbed image. Based on this observation, we propose a double-tail detector that detects an image I as AI-generated if its decoded watermark has a bitwise accuracy larger than τ or smaller than $1-\tau$, i.e., $BA(D(I), w) > \tau$ or $BA(D(I), w) < 1-\tau$. We can calculate the FPR (denoted as $FPR_d(\tau)$) of the double-tail detector with threshold τ as follows:

$$\begin{aligned} FPR_d(\tau) &= \Pr(BA(D(I_o), w) > \tau \text{ or } BA(D(I_o), w) < 1-\tau) \\ &= \Pr(m > n\tau \text{ or } m < n - n\tau) = 2\sum_{k=\lceil n\tau \rceil}^{n} \binom{n}{k} \frac{1}{2^n}, \end{aligned} \quad (2)$$

where $FPR_d(\tau)$ is defined for any original image and the randomness stems from picking the ground-truth watermark w uniformly at random. Therefore, to make $FPR_d(\tau) < \eta$, τ should be at least $\tau^* = \arg\min_\tau 2\sum_{k=\lceil n\tau \rceil}^{n} \binom{n}{k} \frac{1}{2^n} < \eta$. For instance, when $n = 256$ and $\eta = 10^{-4}$, we have $\tau \geq \tau^* \approx 0.621$.

Deployment scenarios: Watermark-based detection of AI-generated content is an emerging topic, and how watermark-based detectors will be deployed in the real-world is still an open question. Nevertheless, we envision the following four deployment scenarios:

Detection-as-a-service. In this scenario, the service provider, who provides the cloud service to generate images, also provides detection-as-a-service to detect its generated images. A user can upload an image to the detection-as-a-service, which returns a binary answer "AI-generated" or "non-AI-generated". In this scenario, the service provider is a computation/communication bottleneck.

End-user detection. In this scenario, the detector is deployed as an end-user application (e.g., a mobile app, a browser plugin), which runs on end-user devices (e.g., smartphone, laptop).

Public detection. In this scenario, the service provider makes its decoder and ground-truth watermark w public so everyone can locally detect images generated by the service provider's AI model. Note that individuals may select their own personalized detection thresholds τ in public detection.

Third-party detection. In this scenario, the service provider shares its decoder and watermark w with selected third parties, so they can locally detect images generated by the service provider's AI model. For instance, OpenAI may share its decoder and watermark with Twitter, so the latter can detect images generated by OpenAI's models that are propagated on Twitter. Note that third parties may select their preferred thresholds τ in third-party detection.

4 THREAT MODEL

Attacker's goal: Suppose an attacker uses the aforementioned cloud service to generate a watermarked image I_w. The attacker aims to *post-process* the watermarked image to evade watermark-based detection while maintaining its visual quality. The attacker may desire to achieve such goals in various scenarios. For instance, the attacker may use the generated image to spread disinformation on the Internet; and the attacker may claim ownership of the AI-generated image. Formally, the attacker aims to turn the watermarked image I_w into a post-processed one I_{pw} via adding a small, human-imperceptible perturbation to it such that a detector falsely predicts I_{pw} as non-AI-generated.

Attacker's background knowledge: Recall that a watermarking method has a ground-truth watermark w, an encoder, and a decoder. A watermark-based detector requires w, the decoder, and a detection threshold τ. Since detection does not involve the encoder, whether it is available to the attacker is not relevant. Nevertheless, we assume the attacker does not have access to the encoder. Since our attack is encoder-agnostic, it is applicable to watermarking methods [9, 38, 41] that embed watermarks to images at generation. Moreover, we assume the attacker does not have access to w. Depending on what information (decoder and/or τ) of the detectors the attacker has access to, we consider the following two scenarios:

White-box. In this threat model, we assume the attacker has white-box access to the decoder of the detectors. This scenario arises in various circumstances: 1) an attacker can directly access the decoder when the service provider makes it public in public detection, e.g., the decoder used by Stable Diffusion is public [26]; 2) an attacker can reverse engineer the end-user application to obtain the decoder when the detector is deployed as an end-user application in end-user detection; 3) a third-party may leak the decoder in third-party detection; and 4) an insider may leak the decoder or an attacker can exploit the computer system vulnerabilities to perform a data leakage attack in detection-as-a-service. A recent example of third-party leakage (not watermarking model, though) is that Meta shared its LLaMA model with verified third parties, one of which leaked it to the public [13].

Note that, given a decoder, different detectors may use different τ. For instance, in public detection (or third-party detection), different individuals (or third-parties) can choose their own τ. Therefore, instead of evading a particular detector with a specific τ, an attacker aims to post-process a watermarked image that can evade detectors with any detection threshold $\tau > 0.5$ in the white-box setting.

Black-box. In this threat model, we assume the attacker has black-box access to a particular detector with a decoder and a τ (called *target detector*), and the attacker aims to evade this target detector. Specifically, the attacker only has access to the binary detection result ("AI-generated" or "non-AI-generated") for any image. This threat model may arise in detection-as-a-service, end-user detection, or third-party detection. For instance, in detection-as-a-service or end-user detection, the attacker can query the target detector to obtain the detection result for any image. In third-party detection, the attacker can also obtain the detection result for any image from a particular third party, e.g., the attacker can upload an image to Twitter and obtain the detection result depending on whether the image is blocked by Twitter or not.

Attacker's capability: In the white-box setting, an attacker can post-process a watermarked image via analyzing the decoder. In the black-box setting, the attacker can query the target detector to

obtain the detection result for any image. Moreover, we assume the attacker can query the target detector multiple times. For instance, the attacker can easily send multiple query images to detection-as-a-service or end-user detection and obtain detection results. We acknowledge that it may take a longer time for the attacker to query a target detector in third-party detection. For instance, when the third-party is Twitter, the attacker uploads a query image to Twitter and may have to wait for some time before obtaining the detection result, i.e., Twitter blocks or does not block the query image. However, as our experiments will show, an attacker only needs dozens of queries to evade a target detector while adding a small perturbation to a watermarked image.

5 OUR WEVADE

5.1 White-box Setting

Suppose we are given a watermarked image I_w and a decoder D. An attacker's goal is to add a small, human-imperceptible perturbation δ to I_w such that the post-processed watermarked image $I_{pw} = I_w + \delta$ evades detectors with any $\tau > 0.5$. We first extend standard adversarial examples to watermarking to find the perturbation δ, which, however, can be defended by the double-tail detector. Then, to address the limitation, we propose a new optimization problem to formulate finding the perturbation δ to evade detection and design an algorithm to solve the optimization problem.

5.1.1 Extending Standard Adversarial Examples to Watermarking (WEvade-W-I). We denote this variant as WEvade-W-I, where W indicates the white-box threat model. The decoder D outputs a watermark, each bit of which can be viewed as a binary class. Therefore, given a watermarked image I_w, one way is to add perturbation δ to it such that D outputs a different binary value for each bit of the watermark. Formally, inspired by the standard adversarial examples [33], we formulate the following optimization problem:

$$\min_{\delta} ||\delta||_\infty$$
$$s.t.\ D(I_w + \delta) = \neg D(I_w), \tag{3}$$

where $||\delta||_\infty$ is the ℓ_∞-norm of the perturbation δ and $\neg$ means flipping each bit of the watermark $D(I_w)$. This optimization problem is hard to solve due to the highly nonlinear constraint. To address the challenge, we reformulate the optimization problem as follows:

$$\min_{\delta} l(D(I_w + \delta), \neg D(I_w)) \tag{4}$$
$$s.t.\ ||\delta||_\infty \leq r,$$
$$D(I_w + \delta) = \neg D(I_w), \tag{5}$$

where l is a loss function to measure the distance between two watermarks and r is a perturbation bound. We discuss more details on solving this reformulated optimization problem in Section 5.1.3.

The loss function should be small when $D(I_w + \delta)$ is close to $\neg D(I_w)$. For instance, the loss function could be ℓ_2 distance, ℓ_1 distance, negative cosine similarity, or average cross-entropy loss. In defining the loss function, we treat $\neg D(I_w)$ as desired "labels". Formally, for ℓ_2 distance, we have $l(D(I_w+\delta), \neg D(I_w)) = \sum_i (F(I_w+\delta)_i - \neg D(I_w)_i)^2$, where $F(I_w+\delta)$ is the second-to-last layer outputs of the decoder neural network D and the subscript i is the index in a vector/bitstring; for ℓ_1 distance, we have $l(D(I_w + \delta), \neg D(I_w)) = \sum_i |F(I_w + \delta)_i - \neg D(I_w)_i|$; and for negative cosine similarity, we have $l(D(I_w + \delta), \neg D(I_w)) = 1 - cos(F(I_w + \delta), \neg D(I_w))$, where we treat $F(I_w + \delta)$ and w_t as vectors and cos is the cosine similarity between them. For cross-entropy loss, we can treat $F(I_w+\delta)_i$ as the possibility that the ith bit is predicted as 1. Then we have $l(D(I_w + \delta), \neg D(I_w)) = -\sum_i (\neg D(I_w)_i \log F(I_w+\delta)_i + (1-\neg D(I_w)_i) \log(1 - F(I_w + \delta)_i))$. We use the second-to-last layer continuous-value outputs instead of the final binary outputs, because the binary outputs are obtained by thresholding the continuous-value outputs (see details in Section 2.2) and thus contain no useful gradient information for updating the perturbation δ.

5.1.2 Formulating a New Optimization Problem (WEvade-W-II). Given a watermarked image I_w, the perturbation δ found by solving the above optimization problem can evade the single-tail detectors with any threshold $\tau > 0.5$. However, our double-tail detector can still detect such post-processed watermarked images because their watermarks have too small bitwise accuracy, as we formally show in our theoretical analysis in Section 6. To address the limitation, we propose a new optimization problem to formulate finding the perturbation δ. Specifically, we aim to find a small perturbation δ such that the decoded watermark $D(I_w + \delta)$ has a bitwise accuracy close to 0.5, compared to the ground-truth watermark w. As a result, the post-processed watermarked image is indistinguishable with original images with respect to bitwise accuracy, evading both single-tail and double-tail detectors. Formally, we formulate finding the perturbation δ as the following optimization problem:

$$\min_{\delta} ||\delta||_\infty \tag{6}$$
$$s.t.\ |BA(D(I_w + \delta), w) - 0.5| \leq \epsilon, \tag{7}$$

where $BA(D(I_w + \delta), w)$ measures the bitwise accuracy of the watermark $D(I_w+\delta)$ compared to the ground-truth one w, ϵ is a small value characterizing the difference between $BA(D(I_w + \delta), w)$ and 0.5, and we call the constraint of the optimization problem *bitwise-accuracy constraint*. However, solving the above optimization problem faces two challenges: 1) the attacker does not have access to the ground-truth watermark w, and 2) the constraint is highly nonlinear, making standard optimization method like *gradient descent (GD)* hard to apply. Next, we discuss how to address the two challenges.

Addressing the first challenge: One way to address the first challenge is to replace the ground-truth watermark w as the watermark $D(I_w)$ decoded from the watermarked image I_w in the optimization problem. However, when the decoded watermark $D(I_w)$ is quite different from w, even if the found perturbation δ satisfies $|BA(D(I_w + \delta), D(I_w)) - 0.5| \leq \epsilon$, there is no formal guarantee that the bitwise-accuracy constraint in Equation 7 is satisfied. To address the challenge, we replace the ground-truth watermark w as a watermark w_t picked uniformly at random, where we call w_t *target watermark*. Moreover, we reformulate the optimization problem such that when the watermark $D(I_w + \delta)$ decoded from the post-processed watermarked image is very close to w_t, it is guaranteed to satisfy the bitwise-accuracy constraint in Equation 7 with high probability. Intuitively, since w_t is picked uniformly at random, it has a bitwise accuracy close to 0.5 compared to any ground-truth watermark w. Therefore, when $D(I_w + \delta)$ is close to w_t, it is likely to have a bitwise accuracy close to 0.5 as well.

Addressing the second challenge: Due to the bitwise-accuracy constraint, it is hard to apply an iterative method like GD. This is because it is hard to find the gradient of δ, moving δ along which can make the bitwise-accuracy constraint more likely to be satisfied. To address this challenge, we reformulate the optimization problem such that it is easier to find a gradient along which δ should be moved. Combining our strategies to address the two challenges, we reformulate the optimization problem as follows:

$$\min_{\delta} l(D(I_w + \delta), w_t) \tag{8}$$

$$s.t.\ ||\delta||_\infty \leq r,$$

$$BA(D(I_w + \delta), w_t) \geq 1 - \epsilon, \tag{9}$$

where l is a loss function to measure the distance between $D(I_w+\delta)$ and w_t, r is a perturbation bound, and ϵ is a small number. Our reformulated optimization problem means that we aim to find a perturbation bounded by r to minimize the loss between $D(I_w + \delta)$ and w_t such that the bitwise accuracy $BA(D(I_w + \delta), w_t)$ is close to 1. Note that a small r may not be able to generate a perturbation δ that satisfies the constraint in Equation 9. Therefore, as detailed in our method to solve the optimization problem, we perform a binary search to find the smallest r such that the found perturbation δ satisfies the constraint in Equation 9.

5.1.3 Solving the Optimization Problems. We propose a unified framework to solve the reformulated optimization problems in WEvade-W-I and WEvade-W-II. Our key idea of solving the reformulated optimization problems is that we use the popular *projected gradient descent (PGD)* [17] to iteratively find the perturbation δ that satisfies the constraints (if possible) for a given r. Then, we perform binary search over r to find the smallest perturbation δ that satisfies the constraints. Specifically, the binary search interval $[r_a, r_b]$ is initialized such that $r_a = 0$ and r_b is a large value (e.g., 2). Then, we pick $r = (r_a + r_b)/2$ and solve a reformulated optimization problem for the given r. If the found perturbation δ satisfies the constraint in the reformulated optimization problem, then we update $r_b = r$, otherwise we update $r_a = r$. We repeat the process until the binary search interval size is smaller than a threshold, e.g., $r_b - r_a \leq 0.001$ in our experiments. Algorithm 1 in Appendix shows our binary search process, where the target watermark $w_t = \neg D(I_w)$ in WEvade-W-I and w_t is a randomly picked watermark in WEvade-W-II. The function FindPerturbation solves a reformulated optimization problem to find δ for a given r.

Next, we discuss the function FindPerturbation, which is illustrated in Algorithm 2 in Appendix. We solve the optimization problem for a given r using PGD. The perturbation δ is initialized to be 0. In each iteration, we compute the gradient of the loss function $l(D(I_w + \delta), w_t)$ with respect to δ and move δ towards the inverse of the gradient by a small step α, which is known as *learning rate*. If the ℓ_∞-norm of δ is larger than the perturbation bound r, we project it so its ℓ_∞-norm is r. We repeat the process for *max_iter* iterations and stop the iterative process early if the constraint in the reformulated optimization problem (i.e., Equation 5 in WEvade-W-I or Equation 9 in WEvade-W-II) is already satisfied.

5.2 Black-box Setting

Surrogate-model-based (WEvade-B-S): The first direction is that the attacker trains a surrogate encoder/decoder, and then performs white-box attacks based on its surrogate decoder. The key hypothesis of such method is that the surrogate detector outputs a similar watermark with the target decoder for a post-processed watermarked image, and thus the post-processed watermarked image constructed to evade the surrogate decoder based detector may also evade the target detector. Specifically, the attacker collects some images and trains an encoder/decoder using the watermarking algorithm on its own images. The attacker's images and the service provider's images used to train encoders/decoders may be from different distributions. After training a surrogate encoder and decoder, the attacker can turn a watermarked image I_w into a post-processed one I_{pw} using the surrogate decoder and the white-box attack, e.g., WEvade-W-II in our experiments. Note that WEvade-B-S does not rely on information of the target detector (e.g., target decoder and τ), and thus the same I_{pw} could be used for all detectors.

Query-based (WEvade-B-Q): WEvade-B-S does not directly take information about the *target detector* into consideration. As a result, the surrogate decoder may be quite different from the target decoder, leading to low evasion rates as shown in our experiments. To address the challenge, WEvade-B-Q finds the post-processed watermarked image I_{pw} by directly querying the target detector. Note that in this setting, we post-process a watermarked image to evade a target detector with a particular threshold τ, unlike the white-box setting where we aim to evade detectors with any threshold $\tau > 0.5$. Finding I_{pw} in such scenario can be viewed as finding adversarial example to the target detector (i.e., a binary classifier) which returns a hard label for a query image. Therefore, we extend state-of-the-art hard-label query-based adversarial example technique called HopSkipJump [6] to find I_{pw} in our problem.

Specifically, HopSkipJump first generates a random initial I_{pw} that evades the target detector by blending the given watermarked image I_w with uniform random noise. Then, HopSkipJump iteratively moves I_{pw} towards I_w to reduce perturbation while always guaranteeing that I_{pw} evades detection. In each iteration, HopSkipJump returns a new I_{pw} and the number of queries to the target detector API used to find such I_{pw}. HopSkipJump stops the iterative process when reaching a given *query budget*. We found that simply applying HopSkipJump to watermark-based detector leads to large perturbations. This is because 1) the random initial I_{pw} may be far away from I_w, and 2) the perturbation may increase after some iterations before reaching the query budget.

Our WEvade-B-S extends HopSkipJump by addressing the two limitations. First, instead of using a random initial I_{pw}, WEvade-B-S uses a post-processed version of I_w as the initial I_{pw}. For instance, we can use JPEG compression to post-process I_w as the initial I_{pw}. In particular, we decrease the quality factor Q of JPEG in the list [99, 90, 70, 50, 30, 10, 1] until finding a post-processed version of I_w that evades detection, which is our initial I_{pw}. When none of the quality factor can generate a post-processed version of I_w that evades the target detector, we revert to the random initial I_{pw} adopted by HopSkipJump. Second, we early stop the iterative process when the perturbation in I_{pw} increases in multiple (denoted as *ES*) consecutive iterations. Algorithm 3 in Appendix shows our WEvade-B-S, where the function HopSkipJump(I_{pw}) returns a new I_{pw} and the number of queries to the API used to find it.

6 THEORETICAL ANALYSIS

Given a watermarked image I_w, our attack turns it into a post-processed watermarked image I_{pw}. We define *evasion rate* of I_{pw} as the probability that it is falsely detected as non-AI-generated, where the randomness (if any) in calculating the probability stems from our attack, e.g., the randomness in picking the target watermark w_t in WEvade-W-II. We formally analyze the evasion rate of WEvade against both single-tail detector and double-tail detector in the white-box and black-box settings. All the proofs are shown in Appendix.

6.1 White-box Setting

WEvade-W-I: Suppose a watermarked image I_w can be correctly detected by a (single-tail or double-tail) detector with threshold $\tau > 0.5$. The following theorem shows that the post-processed watermarked image I_{pw} found by WEvade-W-I is guaranteed to evade the single-tail detector with evasion rate 1, while it is guaranteed to be detected by the double-tail detector (i.e., evasion rate is 0).

Theorem 1. *Given a watermarked image I_w that can be detected by a single-tail or double-tail detector with a threshold $\tau > 0.5$. Suppose I_{pw} is found by our WEvade-W-I. I_{pw} is guaranteed to evade the single-tail detector, but is guaranteed to be detected by the double-tail detector. Formally, we have the following:*

$$\text{Single-tail detector: } BA(D(I_{pw}), w) < \tau, \tag{10}$$

$$\text{Double-tail detector: } BA(D(I_{pw}), w) < 1-\tau \text{ or } BA(D(I_{pw}), w) > \tau, \tag{11}$$

where w is any unknown ground-truth watermark.

WEvade-W-II: The following theorems show the evasion rates of WEvade-W-II against single-tail and double-tail detectors.

Theorem 2. *Given a watermarked image I_w and a single-tail detector with any threshold $\tau > 0.5$. Suppose I_{pw} is found by our WEvade-W-II. For any ground-truth watermark w, the probability (i.e., evasion rate) that I_{pw} successfully evades the single-tail detector can be lower bounded as follows:*

$$Pr(BA(D(I_{pw}), w) \leq \tau) \geq P(\lfloor(\tau - \epsilon)n\rfloor), \tag{12}$$

where n is the watermark length and $P(t) = Pr(m \leq t)$ is the cumulative distribution function of the binomial distribution $m \sim B(n, 0.5)$.

Theorem 3. *Given a watermarked image I_w and a double-tail detector with any threshold $\tau > 0.5$. Suppose I_{pw} is found by our WEvade-W-II. For any ground-truth watermark w, the probability (i.e., evasion rate) that I_{pw} successfully evades the double-tail detector can be lower bounded as follows:*

$$Pr(1-\tau \leq BA(D(I_{pw}), w) \leq \tau) \geq 2P(\lfloor(\tau - \epsilon)n\rfloor) - 1, \tag{13}$$

where n is the watermark length and $P(t) = Pr(m \leq t)$ is the cumulative distribution function of the binomial distribution $m \sim B(n, 0.5)$.

Theorem 2 and 3 indicate that the evasion rate lower bound of a post-processed watermarked image I_{pw} constructed by WEvade-W-II depends on τ used by the detector, ϵ adopted by the attacker in WEvade-W-II, and the watermark length n. For instance, for a detector with a larger τ, the evasion rate is larger.

6.2 Black-box Setting

WEvade-B-S: The evasion rate of WEvade-B-S relies on the "similarity" between the surrogate decoder D' and target decoder D. Based on a formal definition of similarity between the watermarks decoded by the surrogate decoder D' and target decoder D for any image, we can derive the evasion rate of WEvade-B-S. First, we formally define the similarity between D' and D as follows:

Definition 1 ((β,γ)-similar). *Suppose we are given a surrogate decoder D' and target decoder D. We say D' and D are (β,γ)-similar if their outputted watermarks have bitwise accuracy at least β with probability at least γ for an image I picked from the watermarked-image space uniformly at random. Formally, we have:*

$$Pr(BA(D'(I), D(I)) \geq \beta) \geq \gamma. \tag{14}$$

Then, given that D' and D are (β,γ)-similar, the following theorem shows lower bounds of the evasion rates of WEvade-B-S against single-tail detector and double-tail detector.

Theorem 4. *Suppose WEvade-B-S finds an I_{pw} based on a surrogate decoder D'; and D' and the target decoder D are (β,γ)-similar. Then, the evasion rates of I_{pw} for a single-tail detector or double-tail detector with threshold $\tau > 0.5$ are lower bounded as follows:*

Single-tail detector:

$$Pr(BA(D(I_{pw}), w) \leq \tau) \geq \gamma P(\lfloor(\tau + \beta - \epsilon - 1)n\rfloor) \tag{15}$$

Double-tail detector:

$$Pr(1-\tau \leq BA(D(I_{pw}), w) \leq \tau) \geq 2\gamma P(\lfloor(\tau + \beta - \epsilon - 1)n\rfloor) - 1, \tag{16}$$

where w is the unknown ground-truth watermark.

WEvade-B-Q: WEvade-B-Q starts from an initial I_{pw} that evades the target detector. During the iterative process to reduce the perturbation, WEvade-B-Q always guarantees that I_{pw} evades detection. Therefore, the evasion rate of WEvade-B-Q is 1. Note that the evasion rate is only for the target detector.

7 EVALUATION

7.1 Experimental Setup

Datasets: We use three benchmark datasets, including COCO [15], ImageNet [8], and Conceptual Caption (CC) [31]. Following HiDDeN [46] and UDH [42], we randomly sample 10,000 training images from each dataset to train watermarking encoder and decoder. For evaluation, we randomly sample 100 images from the testing set and embed a watermark into each image. For each image in all datasets, we re-scale its size to 128×128.

Post-processing methods: We compare with the following existing post-processing methods, which are widely used to measure robustness of watermarking methods. Each of these post-processing methods has some parameter, which controls the amount of perturbation added to a watermarked image and thus evasion rate.

JPEG. JPEG [43] is a popular image compression method. It has a parameter called *quality factor* Q. A smaller quality factor compresses an image more, is more likely to evade detection, and also adds larger perturbation.

Gaussian noise. This method adds a random Gaussian noise to each pixel of a watermarked image. The Gaussian distribution has

a mean 0 and standard deviation σ. The σ controls the perturbation and thus evasion rate.

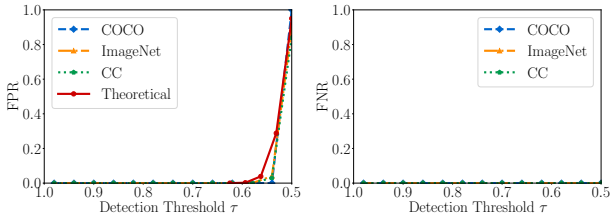
Gaussian blur. This method blurs a watermarked image. It has a parameter called kernel size s and standard deviation σ. We did not observe much impact of the kernel size once it is small enough, and thus we set $s = 5$. However, we will vary σ to control the perturbation added to watermarked images and thus evasion rate.

Brightness/Contrast. This method adjusts the brightness and contrast of an image. Formally, the method has two parameters a and b, where each pixel value x is converted to $ax+b$. b has a smaller impact. We set $b = 0.2$ and vary a to control the perturbation added to watermarked images.

Watermarking methods: We consider two representative learning-based methods HiDDeN [46] and UDH [42], whose implementations are publicly available. To consider watermarks with different lengths, we use 30-bit watermarks in HiDDeN and 256-bit watermarks in UDH. We use the default parameter settings of HiDDeN and UDH in their publicly available code. HiDDeN normalizes the pixel value range [0, 255] to be [-1, 1], while UDH normalizes to [0, 1]. We consider both standard training and adversarial training as described in Section 2.2. but the encoders/decoders are trained using standard training unless otherwise mentioned. In adversarial training, we randomly sample a post-processing method from no post-processing, the existing ones, and ours with a random parameter to post-process each watermarked image in a mini-batch. We use WEvade-W-II with the parameter $\epsilon = 0.01$ if our adversarial post-processing method is sampled. For the existing methods, we consider the following range of parameters during adversarial training: $Q \in [10, 99]$ for JPEG, $\sigma \in [0, 0.1]$ for Gaussian noise, $\sigma \in [0, 1.0]$ for Gaussian blur, and $a \in [1, 5]$ for Brightness/Contrast. We consider these parameter ranges because parameters out of the ranges impact the images' visual quality.

Evaluation metrics: We consider *bitwise accuracy*, *evasion rate*, and *average perturbation*. Bitwise accuracy of an image is the fraction of the bits of its watermark that match with the ground-truth one. Evasion rate is the fraction of post-processed watermarked images that evade detection. Perturbation added to a watermarked image is measured by its ℓ_∞-norm. For each dataset, we report bitwise accuracy, evasion rate, and perturbation averaged over 100 original/watermarked/post-processed testing images. Note that HiDDeN normalizes the pixel value range [0, 255] to be [-1, 1]. Therefore, we divide the perturbation in HiDDeN by 2, so the perturbation represents the fraction of the pixel value range [0, 255] in both HiDDeN and UDH. For instance, a perturbation of 0.02 means changing each pixel value by at most $0.02 * 255 = 5$ of an image.

Parameter settings: We set $max\_iter = 5,000$, $\alpha = 0.1$ for HiDDeN and $\alpha = 1$ for UDH in WEvade-W-I and WEvade-W-II. We use a larger α for UDH because its watermark length is larger. We set $\epsilon = 0.01$ in WEvade-W-II. For WEvade-B-Q, unless otherwise mentioned, we set the query budget to be 2,000 and the early stopping threshold $ES = 5$. By default, we use the ℓ_2-distance as the loss function. Unless otherwise mentioned, we show results when the dataset is COCO, watermarking method is HiDDeN, and detector is the double-tail detector.



(a) FPR (b) FNR

Figure 4: False positive rate (FPR) and false negative rate (FNR) of the double-tail detector based on UDH as the threshold τ varies when there are no attacks.

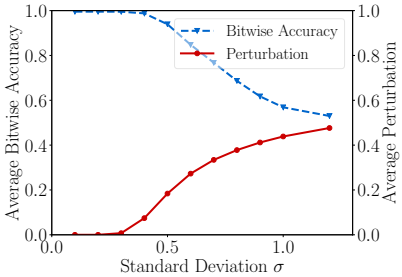
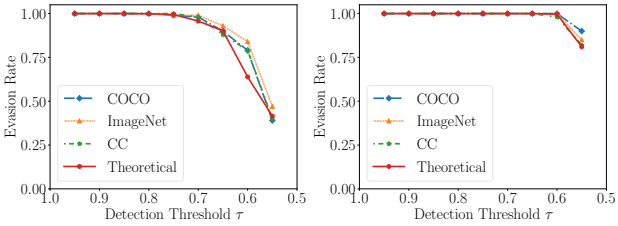


Figure 5: Average bitwise accuracy and average perturbation of the post-processed watermarked images when Gaussian blur uses different standard deviations.

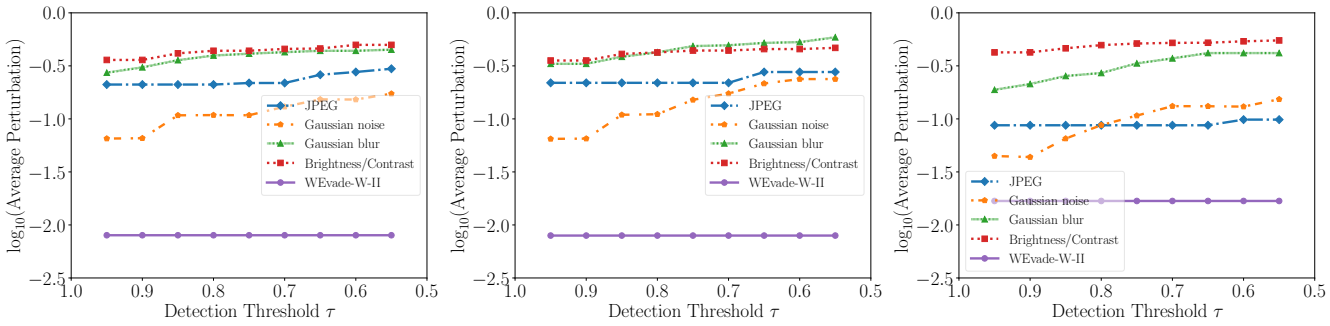


(a) HiDDeN (b) UDH

Figure 6: Evasion rates of WEvade-W-II against the double-tail detector with different τ for the three datasets when the watermarking method is (a) HiDDeN and (b) UDH.

7.2 Detection Results without Attacks

We first show detection results when there are no attacks to post-process watermarked images. Figure 4 shows the false positive rate (FPR) and false negative rate (FNR) of the double-tail detector based on UDH when the threshold τ varies from 0.99 to 0.50, where FPR is the fraction of original testing images that are falsely detected as watermarked and FNR is the fraction of watermarked testing images that are falsely detected as original. The results for the double-tail detector based on HiDDeN and single-tail detector are shown in Figure 19 and Figure 20 in Appendix, respectively. The "Theoretical" curves are the theoretical FPRs of the detectors, i.e., $FPR_s(\tau)$ in Equation 1 and $FPR_d(\tau)$ in Equation 2. There is no theoretical analytical form for FNR, and thus there are no curves corresponding to "Theoretical" in the FNR graphs. Note that $FPR_s(\tau)$ or $FPR_d(\tau)$ is the theoretical FPR for any original image when the ground-truth watermark is picked uniformly at random. More specifically, given any original image, if we pick 100 ground-truth watermarks uniformly at random, the theoretical FPR is roughly the fraction



(a) COCO (b) ImageNet (c) CC

Figure 7: Average perturbation added by each post-processing method to evade the double-tail detector with different threshold τ in the white-box setting. We set the parameters of existing post-processing methods such that they achieve the same evasion rate as our WEvade-W-II. The watermarking method is HiDDeN and the results for UDH are shown in Figure 24 in Appendix.

of the 100 trials in which the original image is falsely detected as watermarked. The empirical FPR shown in Figure 4 can be viewed as estimating the theoretical FPR of each original testing image using one randomly picked ground-truth watermark and then averaging the estimated theoretical FPRs among the original testing images.

We have three observations. First, under no attacks, both single-tail and double-tail detectors are accurate when the threshold τ is set properly. In particular, for HiDDeN (or UDH), both FPR and FNR of both detectors are consistently close to 0 on the three datasets when τ varies from 0.7 to 0.95 (or from 0.6 to 0.99). The range of such τ is wider for UDH than for HiDDeN, i.e., [0.6, 0.99] vs. [0.7, 0.95]. This is because UDH uses a longer watermark than HiDDeN, i.e., 256 vs. 30 bits. Second, the theoretical FPR is close to the empirical FPRs, i.e., the "Theoretical" curve is close to the other three FPR curves in a graph. They do not exactly match because the empirical FPRs are estimated using only one randomly picked ground-truth watermark. Third, given the same threshold τ, the double-tail detector has a higher FPR than the single-tail detector, which is more noticeable when τ is small (e.g., 0.55). This is because the double-tail detector considers both the left and right tails of the bitwise-accuracy distribution (see illustration in Figure 3).

7.3 Attack Results in the White-box Setting

WEvade outperforms existing post-processing methods: Each existing post-processing method has a parameter (discussed in Section 7.1), which controls how much perturbation is added to a watermarked image. Figure 5 shows the average bitwise accuracy and average perturbation of the watermarked images post-processed by Gaussian blur with different parameter values, where HiDDeN and COCO dataset are used. Figure 22 and Figure 23 in Appendix show the results on other post-processing methods and datasets. Based on these results, we compare WEvade with existing post-processing methods with respect to evasion rate and average perturbation added to the watermarked images. Note that there exists a trade-off between evasion rate and average perturbation. Therefore, for a given threshold τ, we tune the parameters of the existing methods such that they achieve similar evasion rates (within 1% difference) with WEvade and we compare the average perturbation.

Figure 6 shows the evasion rates of WEvade-W-II when the double-tail detector uses different threshold τ, while Figure 7 shows

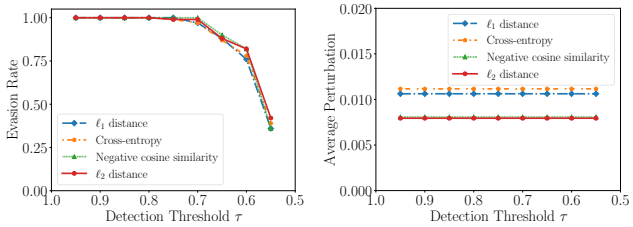


Figure 8: Comparing different loss functions.

the average perturbations that each method requires to achieve such evasion rates. The "Theoretical" curves in Figure 6 correspond to the theoretical lower bounds of evasion rates of WEvade-W-II in Theorem 3, i.e., $2P(\lfloor(\tau-\epsilon)n\rfloor)-1$. Specifically, $\epsilon = 0.01$ and $n = 30$ in our experiments and we use $2P(\lfloor(\tau-\epsilon)n\rfloor)-1$ to calculate the lower bound of evasion rate for any τ. The average perturbation of WEvade-W-II is a straight line in Figure 7 because the perturbation added by WEvade-W-II does not depend on τ. Note that, in our experiments, we give advantages to existing post-processing methods, i.e., we assume they can tune their parameters for a given threshold τ, while our WEvade-W-II does not assume the knowledge of τ.

First, the empirical evasion rates are close to the "Theoretical" lower bounds in Figure 6, which validates our theoretical analysis. The empirical evasion rates are sometimes slightly lower than the theoretical lower bounds because the empirical evasion rates are calculated using a small number (100 in our experiments) of watermarked images. Second, our results show that WEvade-W-II substantially outperforms existing post-processing methods. In particular, WEvade-W-II requires much smaller perturbations to achieve high evasion rates. We also found that when existing methods use parameter values to achieve average perturbations no more than WEvade-W-II, their evasion rates are all 0.

Comparing WEvade-W-I and WEvade-W-II: Figure 21 in Appendix shows the evasion rates and average perturbations of WEvade-W-I and WEvade-W-II as the single-tail detector or double-tail detector uses different threshold τ, where the dataset is COCO and watermarking method is HiDDeN. First, we observe that WEvade-W-I achieves evasion rate of 1 for the single-tail detector while 0 for the double-tail detector, which is consistent with our Theorem 1. Second, for the single-tail detector, WEvade-W-I achieves higher

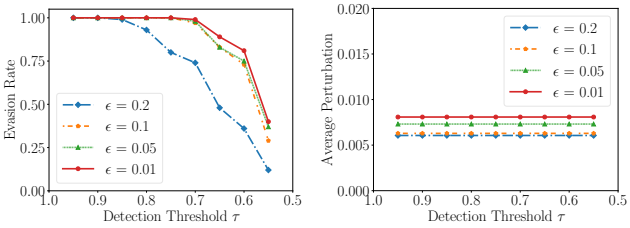


Figure 9: Comparing different ϵ values.

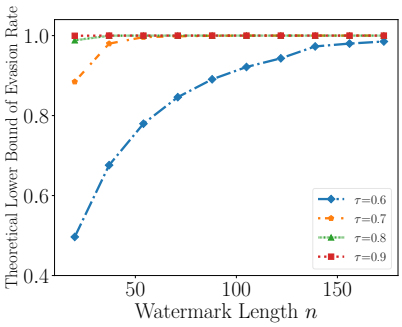


Figure 10: Impact of watermark length n.

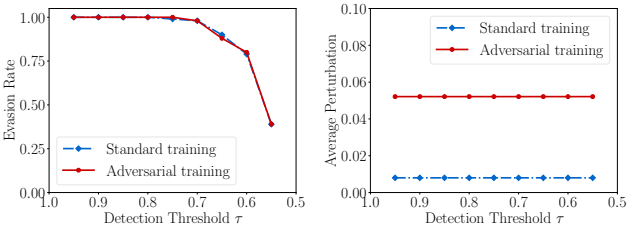
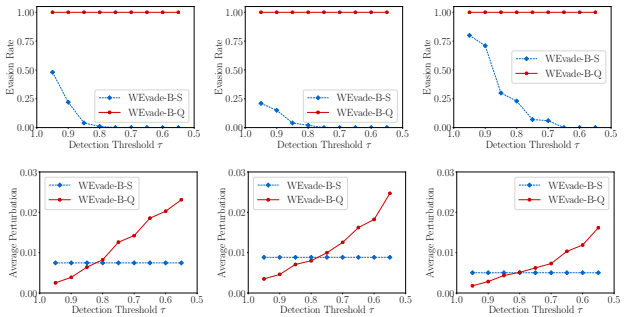


Figure 11: Standard vs. adversarial training for WEvade-W-II.



(a) COCO **(b) ImageNet** **(c) CC**

Figure 12: Comparing evasion rates (*first row*) and average perturbations (*second row*) of WEvade-B-S and WEvade-B-Q in the black-box setting. The watermarking method is HiDDeN and Figure 26 in Appendix shows results for UDH.

evasion rates than WEvade-W-II when τ is small (e.g., 0.6) but incurs larger average perturbation than WEvade-W-II. This is because WEvade-W-I adds (larger) perturbation to flip each bit of the watermark of the watermarked image. However, we stress that their average perturbations are both very small. Third, for the double-tail detector, WEvade-W-II achieves higher evasion rates and incurs smaller average perturbations than WEvade-W-I. Note that the perturbations added by both WEvade-W-I and WEvade-W-II do not depend on the detector, and thus the average-perturbation curves for WEvade-W-I (or WEvade-W-II) are the same for the single-tail detector and double-tail detector in Figure 21.

Impact of loss function: Figure 8 compares different loss functions with respect to evasion rate and average perturbation of WEvade-W-II. We observe that these loss functions achieve comparable results, though ℓ_2-distance and negative cosine similarity achieve slightly smaller average perturbations. The reason is that, in our Algorithm 1, we find the smallest perturbation that satisfies the constraint in Equation 9 no matter what loss function is used; and in Algorithm 2, we early stop as long as the constraint in Equation 9 is satisfied. Moreover, our Theorem 3 shows that the evasion rate of WEvade-W-II does not depend on the loss function once the found perturbation satisfies the constraint in Equation 9.

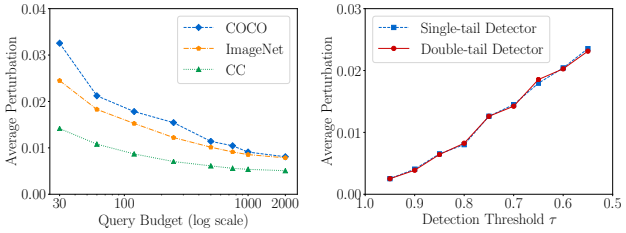
Impact of ϵ: Figure 9 compares different ϵ values with respect to evasion rate and average perturbation of WEvade-W-II. We observe that ϵ achieves a trade-off between evasion rate and average perturbation. As ϵ increases, perturbation decreases because Equation 9 is easier to be satisfied; but evasion rate also decreases because the decoded watermark is less similar to the target watermark w_t.

Impact of watermark length n: Figure 10 shows the theoretical lower bound of evasion rate of WEvade-W-II to double-tail detector (i.e., $2P(\lfloor(\tau-\epsilon)n\rfloor)-1$) as a function of the watermark length n, where $\epsilon = 0.01$ and τ varies from 0.6 to 0.9. We observe that the lower bound increases as n increases. This is because the randomly picked target watermark w_t is more likely to have a bitwise accuracy 0.5 compared to the ground-truth watermark as n increases.

Adversarial training improves robustness but is still insufficient: Figure 11 compares standard training and adversarial training with respect to the evasion rates and average perturbations of WEvade-W-II. We have three observations. First, adversarial training improves robustness of the detector. In particular, WEvade-W-II achieves the same evasion rates for standard and adversarial training. This is because evasion rates of WEvade-W-II do not depend on how the encoder and decoder are trained. However, WEvade-W-II needs to add larger perturbations on average when adversarial training is used. Second, adversarial training is still insufficient. Specifically, the perturbations added by WEvade-W-II are still small, which maintain visual quality of the images well (Figure 1 shows some example images). Third, WEvade-W-II still outperforms existing post-processing methods when adversarial training is used. In particular, Figure 25 in Appendix shows that WEvade-W-II still adds much smaller perturbations than existing methods when they tune parameters to achieve similar evasion rates with WEvade-W-II.

7.4 Attack Results in the Black-box Setting

WEvade-B-S vs. WEvade-B-Q: Figure 12 shows the evasion rate and average perturbation of WEvade-B-S and WEvade-B-Q on the three datasets. Note that, for target detectors with different τ, we apply WEvade-B-Q separately to find the (different) perturbations for a watermarked image, while WEvade-B-S adds τ-agnostic perturbation to a watermarked image. First, WEvade-B-Q always achieves evasion rate of 1 while the evasion rate of WEvade-B-S decreases to 0 as the threshold τ decreases. This is because the surrogate



(a) Impact of query budget max_q (b) Single-tail vs. double-tail detector

Figure 13: (a) Average perturbation of WEvade-B-Q as query budget varies. (b) Average perturbation of WEvade-B-Q to evade the single-tail detector or double-tail detector with different threshold τ.

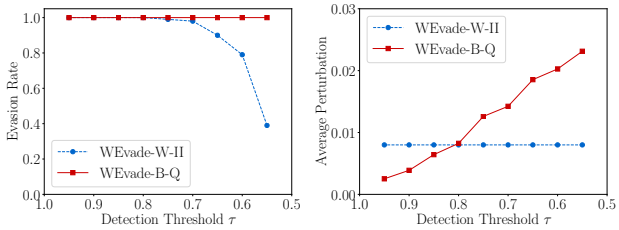
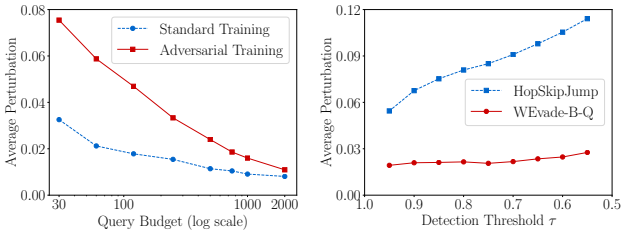


Figure 14: White-box vs. black-box.



(a) Standard vs. adversarial training (b) WEvade-B-Q vs. HopSkipJump

Figure 15: (a) Average perturbation of WEvade-B-Q as the query budget increases. (b) WEvade-B-Q vs. HopSkipJump.

decoder and the target decoder output dissimilar watermarks for an image. As our Theorem 4 shows, when the surrogate decoder and the target decoder are more likely to output dissimilar watermarks, the evasion rate of WEvade-B-S decreases. Second, WEvade-B-Q adds larger perturbation as τ decreases. This is because the decision boundary of a detector with smaller τ is further away from the watermarked images and WEvade-B-Q requires larger perturbations to move them across such boundary. Third, the perturbation of WEvade-B-S does not depend on τ because it uses the white-box attack WEvade-W-II to find perturbations.

Impact of the number of queries on WEvade-B-Q: Figure 13a shows the average perturbation added by WEvade-B-Q when the query budget max_q per watermarked image varies, where the threshold $\tau = \tau^* = 0.83$ (corresponding to FPR=10^{-4}). Note that the evasion rate is always 1. We observe that the average perturbation added by WEvade-B-Q decreases rapidly as the query budget increases. Moreover, when the query budget is small, the average perturbation is already small. For instance, when the query budget is 30 and dataset is COCO, the average perturbation added by WEvade-B-Q is 0.032. On the contrary, existing post-processing methods JPEG, Gaussian noise, Gaussian blur, and Brightness/Contrast respectively add average perturbations 0.211, 0.109, 0.395, and 0.439 to achieve evasion rates close to 1. We acknowledge that WEvade-B-Q requires queries for each watermarked image, so the total number of queries may be large when an attacker aims to evade detection of many watermarked images. However, we note that an attacker can perform a high-profile targeted attack by evading detection of a single or a small number of watermarked images, e.g., a fake image of Elon Musk dating GM CEO Mary Barra [32]. In such scenarios, an attacker can afford a larger number of queries for the targeted watermarked images.

Single-tail vs. double-tail detector: Figure 13b shows the average perturbations added by WEvade-B-Q to evade the single-tail detector and double-tail detector. We observe that WEvade-B-Q adds similar perturbations to evade the two detectors. The reason is that WEvade-B-Q only uses the detector API without considering the internal mechanisms of the detector. Note that the evasion rates of WEvade-B-Q are always 1.

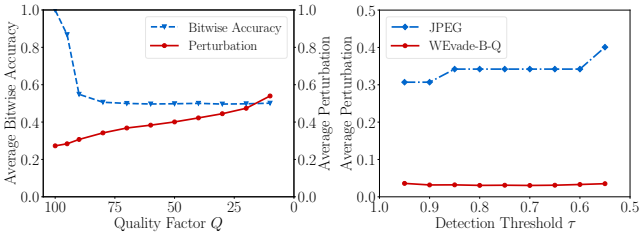
Black-box vs. white-box: Figure 14 compares the evasion rate and average perturbation of WEvade in the white-box (i.e., WEvade-W-II) and black-box settings (i.e., WEvade-B-Q). First, WEvade-B-Q adds smaller perturbations when τ is large (e.g., 0.9) but larger perturbations when τ is small (e.g., 0.6). This is because WEvade-B-Q requires larger perturbations to move watermarked images across the decision boundary of a detector with smaller τ while WEvade-W-II is agnostic to τ. However, we stress that the perturbations of both WEvade-B-Q and WEvade-W-II are small. Second, the perturbations of WEvade-B-Q are still much smaller than those of existing post-processing methods (refer to Figure 7). Third, WEvade-B-Q achieves higher evasion rates than WEvade-W-II when τ is small (e.g., 0.6). This is because WEvade-B-Q guarantees evasion rate of 1.

Adversarial training: Figure 15a compares the average perturbations added by WEvade-B-Q with different query budget max_q for detectors obtained by standard training and adversarial training, where we set $\tau = \tau^* = 0.83$ (corresponding to FPR=10^{-4}). Adversarial training improves robustness in the sense that an attacker needs more queries to achieve similar level of perturbation. However, we stress that adversarial training is insufficient because a moderate number of queries can still achieve small perturbations.

Comparing WEvade-B-Q with HopSkipJump: Figure 15b compares WEvade-B-Q with HopSkipJump in terms of average perturbations, where the watermarking method is UDH and dataset is COCO. We observe that WEvade-B-Q adds much smaller perturbations than HopSkipJump. This is because WEvade-B-Q uses JPEG compressed version of a watermarked image as initialization and adopts early stopping when the added perturbation increases. Figure 27 in Appendix further shows that both the initialization and early stopping contribute to WEvade-B-Q. We note that WEvade-B-Q achieves comparable perturbations with HopSkipJump for HiDDeN. This is because HiDDeN uses 30-bit watermarks and thus the detectors have much simpler decision boundaries.

7.5 Attacking Stable Diffusion's Detector

We generate 100 watermarked images using Stable Diffusion with default setting. We use *sd-v1-1.ckpt* as the checkpoint. Stable Diffusion uses a watermark="StableDiffusionV1", which is represented



(a) JPEG (b) Average Perturbation

Figure 16: (a) Average bitwise accuracy and average perturbation of the Stable Diffusion watermarked images post-processed by JPEG with different quality factor Q. (b) Average perturbation added by JPEG compression and WEvade-B-Q to evade the double-tail detector with different threshold τ.

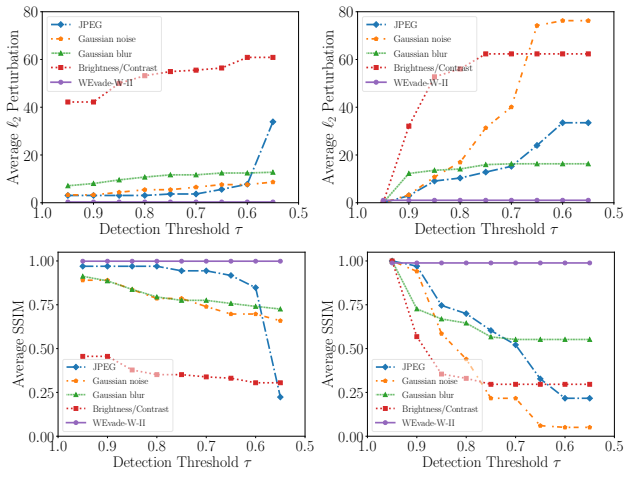
(a) Watermarked (b) JPEG (c) WEvade-B-Q

Figure 17: Illustration of a Stable Diffusion watermarked image and the versions post-processed by JPEG and WEvade-B-Q to evade watermark-based detection.

as 136 bits. The decoder can decode the exact watermark from each of the 100 watermarked images. Figure 16a shows the average bitwise accuracy and average perturbation of the watermark images post-processed by JPEG with different quality factor Q. When Q is around 80, the bitwise accuracy already reduces to be around 0.5, which means a watermark-based detector cannot distinguish JPEG compressed watermarked images with original images. Figure 16b shows the average perturbation incurred by JPEG compression and WEvade-B-Q to evade the double-tail detector. Our WEvade-B-Q incurs much smaller perturbations than JPEG compression. Figure 17 shows an example Stable Diffusion watermarked image, its JPEG compressed version, and the version post-processed by WEvade-B-Q to evade the double-tail detector with $\tau = 0.66$ (corresponding to FPR=10^{-4}). As we can see, both JPEG compression and WEvade-B-Q can evade the Stable Diffusion's detector, which is based on a non-learning-based watermarking method, without sacrificing the image quality.

8 DISCUSSION AND LIMITATIONS

Other metrics to quantify perturbation: Attacker's goal is to add small perturbation to evade detection while preserving visual quality of the image. We use ℓ_∞-norm of the perturbation to quantify whether it preserves visual quality, which is a popular choice in adversarial examples [5, 12]. In particular, when ℓ_∞-norm of the perturbation is small enough, the visual quality is preserved. We can also use other ℓ_p-norms, e.g., ℓ_2-norm, or SSIM [36] between a watermarked image and its post-processed version, to quantify the perturbation. For instance, Figure 18 compares the perturbations added by different post-processing methods in the white-box setting when using ℓ_2-norm or SSIM to quantify the perturbation, while Figure 28 in Appendix shows the results in the black-box setting,



(a) Standard training (b) Adversarial training

Figure 18: Average perturbation, measured by ℓ_2-norm (first row) or SSIM (second row), added by each post-processing method to evade the double-tail detector with different threshold τ in the white-box setting. We set the parameters of existing post-processing methods such that they achieve the same evasion rate as our WEvade-W-II. The watermarking method is HiDDeN and dataset is COCO.

where WEvade uses the default parameter settings described in Section 7.1. Our results show that WEvade still adds much smaller perturbations than existing methods when ℓ_2-norm or SSIM is used to quantify the perturbation. We acknowledge that ℓ_p-norms and SSIM are approximate measures of perturbations' impact on visual quality. Previous works [30] on adversarial examples showed that small ℓ_p-norms of perturbations may not be sufficient nor necessary conditions to maintain visual quality. It is an interesting future work to explore other metrics to quantify the impact of perturbation on visual quality specifically in the generative AI domain.

Provably robust watermarking methods: The fundamental reason that watermarking-based detectors can be evaded by our attack is that existing watermarking methods do not have provable robustness guarantees. Specifically, an attacker can add a small perturbation to a watermarked image such that the decoder outputs a different watermark for the post-processed watermarked image. To defend against such attacks, one interesting future work is to build watermarking methods with provable robustness guarantees. In particular, a provably robust watermarking method is guaranteed to output similar watermarks for a watermarked image and its post-processed version once the added perturbation is bounded, e.g., its ℓ_∞-norm or ℓ_2-norm is smaller than a threshold. For instance, if the watermarks decoded from a watermarked image and its post-processed version are guaranteed to have bitwise accuracy of 0.85 once the ℓ_∞-norm of the perturbation is bounded by 0.03, then a detector with threshold $\tau = 0.8$ is guaranteed to detect the post-processed version once the ℓ_∞-norm of the perturbation is bounded by 0.03. If the perturbation bound is large enough to be human-perceptible, an attacker has to sacrifice visual quality of the watermarked image in order to evade watermarking-based detector, leading to a dilemma for the attacker, i.e., either being detected or perturbed images have low quality.

9 CONCLUSION AND FUTURE WORK

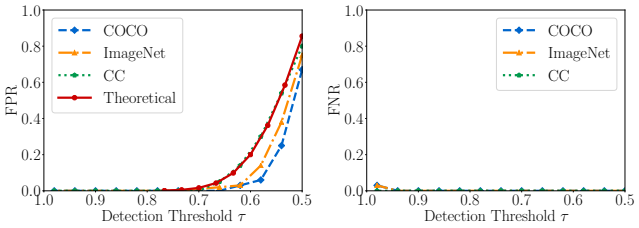
We find that watermark-based detection of AI-generated content is vulnerable to strategic, adversarial post-processing. An attacker can add a small, human-imperceptible perturbation to an AI-generated, watermarked image to evade detection. Our results indicate that watermark-based AI-generated content detection is not as robust as previously thought. We also find that simply extending standard adversarial examples to watermarking is insufficient since they do not take the unique characteristics of watermarking into consideration. An interesting future work is to explore watermark-based detectors with provable robustness guarantees.

ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their constructive comments. This work was supported by NSF grant No. 1937787, 1937786, 2112562, and 2125977, as well as ARO grant No. W911NF2110182.

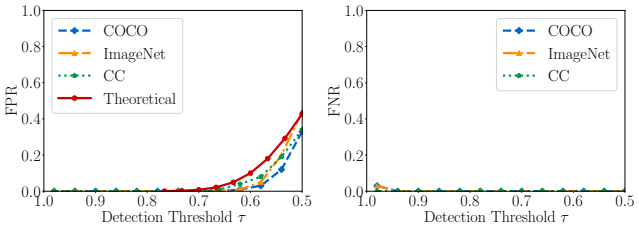
REFERENCES

[1] Sahar Abdelnabi and Mario Fritz. 2021. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *IEEE Symposium on Security and Privacy*.
[2] ARTnews. 2023. US Copyright Office: AI Generated Works Are Not Eligible for Copyright. https://www.artnews.com/art-news/news/ai-generator-art-text-us-copyright-policy-1234661683.
[3] Ning Bi, Qiyu Sun, Daren Huang, Zhihua Yang, and Jiwu Huang. 2007. Robust image watermarking based on multiband wavelets and empirical mode decomposition. *IEEE Transactions on Image Processing* (2007).
[4] Xiaoyu Cao and Neil Zhenqiang Gong. 2022. Understanding the security of deepfake detection. In *EAI International Conference on Digital Forensics and Cyber Crime*.
[5] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*.
[6] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. 2020. Hopskipjumpattack: A query-efficient decision-based attack. In *IEEE Symposium on Security and Privacy*.
[7] Riccardo Corvi, Davide Cozzolino, Giada Zingarini, Giovanni Poggi, Koki Nagano, and Luisa Verdoliva. 2022. On the detection of synthetic images generated by diffusion models. *arXiv preprint arXiv:2211.00680* (2022).
[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
[9] Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. 2023. The Stable Signature: Rooting Watermarks in Latent Diffusion Models. *arXiv preprint arXiv:2303.15435* (2023).
[10] Joel Frank, Thorsten Eisenhofer, Lea Schönherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. 2020. Leveraging frequency analysis for deep fake image recognition. In *International Conference on Machine Learning*.
[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* (2020).
[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
[13] James Vincent. 2023. Meta's powerful AI language model has leaked online. https://www.theverge.com/2023/3/8/23629362/meta-ai-language-model-llama-leak-online-misuse.
[14] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. *arXiv preprint arXiv:2301.10226* (2023).
[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*.
[16] Xiyang Luo, Ruohan Zhan, Huiwen Chang, Feng Yang, and Peyman Milanfar. 2020. Distortion agnostic deep watermarking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
[17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).
[18] Makena Kelly. 2023. Meta, Google, and OpenAI promise the White House they'll develop AI responsibly. https://www.theverge.com/2023/7/21/23802274/artificial-intelligence-meta-google-openai-white-house-security-safety.
[19] MARKETSANDMARKETS. 2023. Generative AI Market. https://www.marketsandmarkets.com/Market-Reports/generative-ai-market-142870584.html.
[20] Marking the Photo. 2022. How to Remove Dall-E Watermark. https://www.youtube.com/watch?v=6EMROCxGCIA.
[21] Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *arXiv preprint arXiv:2301.11305* (2023).
[22] OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. https://openai.com/blog/chatgpt.
[23] Shelby Pereira and Thierry Pun. 2000. Robust template matching for affine resistant image watermarks. *IEEE Transactions on Image Processing* (2000).
[24] Qingquan Wang and buley. 2020. Invisible watermark. https://github.com/ShieldMnt/invisible-watermark.
[25] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *International Conference on Machine Learning*.
[26] Robin Rombach. 2022. Stable Diffusion watermark decoder. https://github.com/CompVis/stable-diffusion/blob/main/scripts/tests/test_watermark.py.
[27] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
[28] Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can AI-Generated Text be Reliably Detected? *arXiv preprint arXiv:2303.11156* (2023).
[29] Zeyang Sha, Zheng Li, Ning Yu, and Yang Zhang. 2022. DE-FAKE: Detection and Attribution of Fake Images Generated by Text-to-Image Diffusion Models. *arXiv preprint arXiv:2210.06998* (2022).
[30] Mahmood Sharif, Lujo Bauer, and Michael K Reiter. 2018. On the suitability of lp-norms for creating and preventing adversarial examples. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*.
[31] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Annual Meeting of the Association for Computational Linguistics*.
[32] Shivdeep Dhaliwal. 2023. Elon Musk isn't dating GM's Mary Barra: he has this to say though on the photos. https://www.benzinga.com/news/23/03/31505898/elon-musk-isnt-dating-gms-mary-barra-he-has-this-to-say-though-on-the-photos.
[33] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).
[34] Matthew Tancik, Ben Mildenhall, and Ren Ng. 2020. Stegastamp: Invisible hyperlinks in physical photographs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
[35] Sheng-Yu Wang, Oliver Wang, Andrew Owens, Richard Zhang, and Alexei A Efros. 2019. Detecting photoshopped faces by scripting photoshop. In *IEEE/CVF International Conference on Computer Vision*.
[36] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* (2004).
[37] Bingyang Wen and Sergul Aydore. 2019. Romark: A robust watermarking system using adversarial training. *arXiv preprint arXiv:1910.01221* (2019).
[38] Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. 2023. Tree-Ring Watermarks: Fingerprints for Diffusion Images that are Invisible and Robust. *arXiv preprint arXiv:2305.20030* (2023).
[39] Yuankun Yang, Chenyue Liang, Hongyu He, Xiaoyu Cao, and Neil Zhenqiang Gong. 2021. Faceguard: Proactive deepfake detection. *arXiv preprint arXiv:2109.05673* (2021).
[40] Ning Yu, Larry S Davis, and Mario Fritz. 2019. Attributing fake images to gans: Learning and analyzing gan fingerprints. In *IEEE/CVF International Conference on Computer Vision*.
[41] Ning Yu, Vladislav Skripniuk, Sahar Abdelnabi, and Mario Fritz. 2021. Artificial fingerprinting for generative models: Rooting deepfake attribution in training data. In *IEEE/CVF International Conference on Computer Vision*.
[42] Chaoning Zhang, Philipp Benz, Adil Karjauv, Geng Sun, and In So Kweon. 2020. Udh: Universal deep hiding for steganography, watermarking, and light field messaging. *Advances in Neural Information Processing Systems* (2020).
[43] Chaoning Zhang, Adil Karjauv, Philipp Benz, and In So Kweon. 2020. Towards robust data hiding against (jpeg) compression: A pseudo-differentiable deep learning approach. *arXiv preprint arXiv:2101.00973* (2020).
[44] Kevin Alex Zhang, Alfredo Cuesta-Infante, Lei Xu, and Kalyan Veeramachaneni. 2019. SteganoGAN: High capacity image steganography with GANs. *arXiv preprint arXiv:1901.03892* (2019).
[45] Hanqing Zhao, Wenbo Zhou, Dongdong Chen, Tianyi Wei, Weiming Zhang, and Nenghai Yu. 2021. Multi-attentional deepfake detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.
[46] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. 2018. Hidden: Hiding data with deep networks. In *European Conference on Computer Vision*.

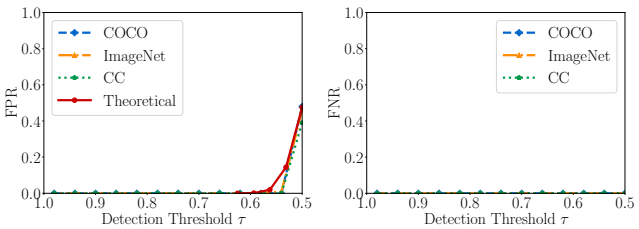


(a) FPR (b) FNR

Figure 19: FPR and FNR of the double-tail detector based on HiDDeN as the threshold τ varies when there are no attacks to post-process the watermarked images.



(a) HiDDeN, FPR (b) HiDDeN, FNR



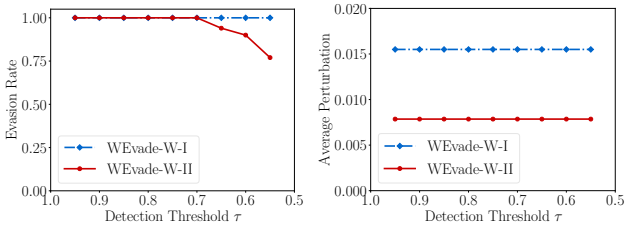
(c) UDH, FPR (d) UDH, FNR

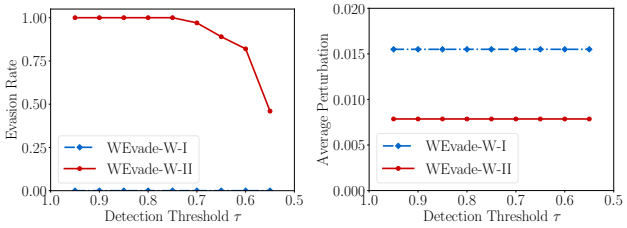
Figure 20: FPR and FNR of the single-tail detector as the threshold τ varies when there are no attacks to post-process the watermarked images.

Algorithm 1 WEvade-W-I and WEvade-W-II

Input: Watermarked image I_w and target watermark w_t
Output: Post-processed watermarked image I_{pw}

$r_b \leftarrow 2$
$r_a \leftarrow 0$
while $r_b - r_a > 0.001$ **do**
  $r \leftarrow (r_a + r_b)/2$
  $\delta' \leftarrow$ FindPerturbation (I_w, w_t, r)
  if ((WEvade-W-I & Equation 5 is satisfied) or (WEvade-W-II & Equation 9 is satisfied)) **then**
    $r_b \leftarrow r$
    $\delta \leftarrow \delta'$
  else
    $r_a \leftarrow r$
  end if
end while
return $I_w + \delta$





(a) Evasion rate (b) Average perturbation

Figure 21: Comparing WEvade-W-I with WEvade-W-II against the single-tail (*first row*) and double-tail (*second row*) detector.

Algorithm 2 FindPerturbation (I_w, w_t, r)

Input: Decoder D, objective function l, learning rate α, and maximum number of iterations *max_iter*.
Output: Perturbation δ

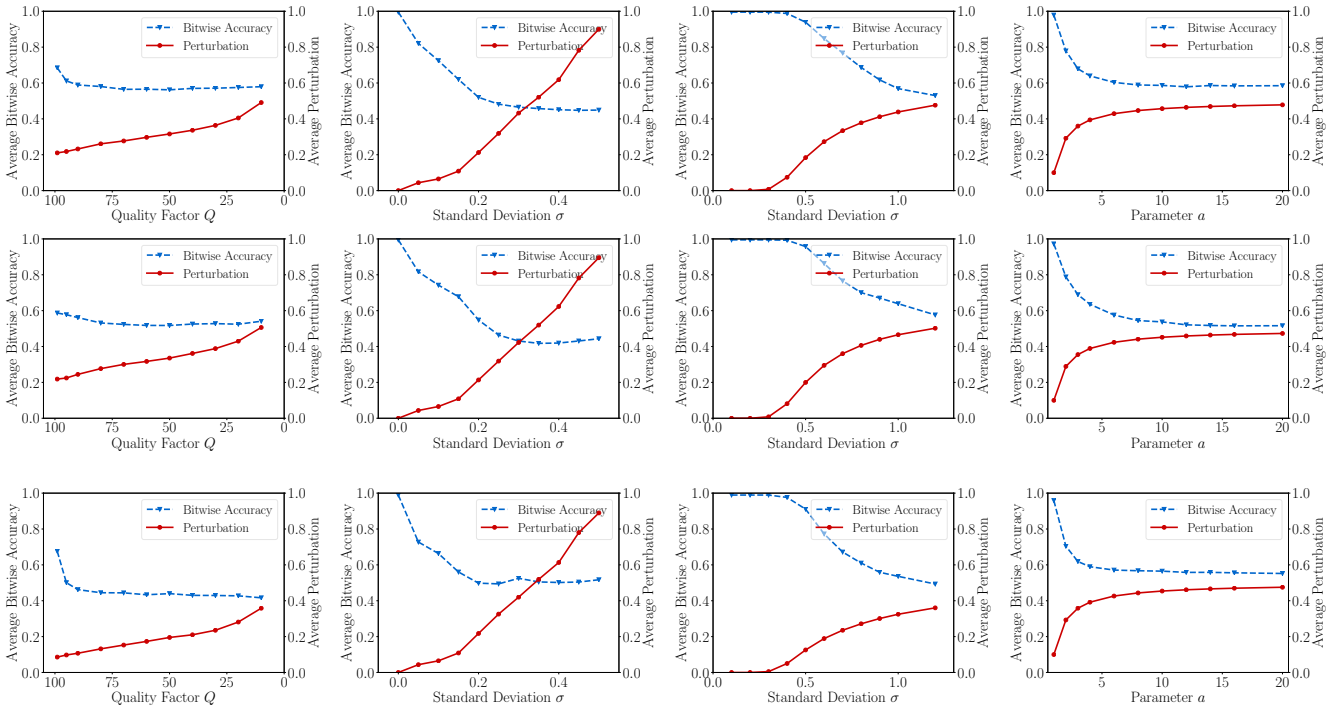
$\delta \leftarrow 0$
for $k = 1$ to *max_iter* **do**
  $g \leftarrow \nabla_\delta l(D(I_w + \delta), w_t)$
  $\delta \leftarrow \delta - \alpha \cdot g$
  //Projection to satisfy the perturbation bound
  if $\|\delta\|_\infty > r$ **then**
    $\delta \leftarrow \delta \cdot \frac{r}{\|\delta\|_\infty}$
  end if
  //Early stopping
  if ((WEvade-W-I & Equation 5 is satisfied) or (WEvade-W-II & Equation 9 is satisfied)) **then**
    return δ
  end if
end for
return δ

A PROOF OF THEOREM 1

For the standard detector, I_w is correctly detected and thus we have $BA(D(I_w), w) > \tau > 0.5$. Therefore, we have:

$$\begin{aligned} &BA(D(I_{pw}), w) \\ &= BA(\neg D(I_w), w) = 1 - BA(D(I_w), w) \\ &< 1 - \tau < \tau. \end{aligned}$$

For the adaptive detector, I_w is correctly detected and thus we have $BA(D(I_w), w) > \tau$ or $BA(D(I_w), w) < 1 - \tau$, where $\tau > 0.5$. Since $BA(D(I_{pw}), w) = 1 - BA(D(I_w), w)$, we have $BA(D(I_{pw}), w) > \tau$ or $BA(D(I_{pw}), w) < 1 - \tau$.



(a) JPEG (b) Gaussian noise (c) Gaussian blur (d) Brightness/Contrast

Figure 22: Average bitwise accuracy and average perturbation of the post-processed watermarked images when an existing post-processing method uses different parameter values. The watermarking method is HiDDeN. The datasets are COCO (first row), ImageNet (second row), and CC (third row).

B PROOF OF THEOREM 2

We denote $D(I_{pw}) = w_{I_{pw}}$. According to Equation 9, we have:

$$BA(w_{I_{pw}}, w_t) = 1 - \frac{|w_{I_{pw}} - w_t|_1}{n} \geq 1 - \epsilon,$$
$$\implies |w_{I_{pw}} - w_t|_1 \leq \epsilon n,$$

where $|\cdot|_1$ is ℓ_1 distance between two binary vectors. Then, according to the triangle inequality, we have:

$$\begin{aligned}|w_t - w|_1 &= |w_t - w_{I_{pw}} + w_{I_{pw}} - w|_1 \\ &\leq |w_t - w_{I_{pw}}|_1 + |w_{I_{pw}} - w|_1 \\ &\leq \epsilon n + |w_{I_{pw}} - w|_1.\end{aligned}$$

Therefore, we have:

$$\begin{aligned}&\Pr(BA(D(I_{pw}), w) \leq \tau) \\ &= \Pr(BA(w_{I_{pw}}, w) \leq \tau) \\ &= \Pr(1 - \frac{|w_{I_{pw}} - w|_1}{n} \leq \tau) \\ &= \Pr(|w_{I_{pw}} - w|_1 \geq (1-\tau)n) \\ &\geq \Pr(|w_t - w|_1 - \epsilon n \geq (1-\tau)n) \\ &= \Pr(|w_t - w|_1 \geq (1 - \tau + \epsilon)n),\end{aligned}$$

Since w_t is picked uniformly at random, we know $|w_t - w|_1$ follows a *binomial distribution*, i.e., $|w_t - w|_1 \sim B(n, 0.5)$. Thus, we have:

$$\begin{aligned}&\Pr(BA(D(I_{pw}), w) \leq \tau) \\ &\geq \Pr(|w_t - w|_1 \geq \lceil (1 - \tau + \epsilon)n \rceil) \\ &= P(\lfloor (\tau - \epsilon)n \rfloor),\end{aligned}$$

where $P(t) = \Pr(m \leq t)$ is the cumulative distribution function of the binomial distribution $m \sim B(n, 0.5)$.

C PROOF OF THEOREM 3

According to Equation 9, we have:

$$BA(w_{I_{pw}}, w_t) = 1 - \frac{|w_{I_{pw}} - w_t|_1}{n} \leq 1 - \epsilon,$$
$$\implies |w_{I_{pw}} - w_t|_1 \leq \epsilon n.$$

Then, according to the triangle inequality, we have:

$$\begin{aligned}|w_{I_{pw}} - w|_1 &= |w_{I_{pw}} - w_t + w_t - w|_1 \\ &\leq |w_{I_{pw}} - w_t|_1 + |w_t - w|_1 \leq \epsilon n + |w_t - w|_1.\end{aligned}$$

Similarly, we have:

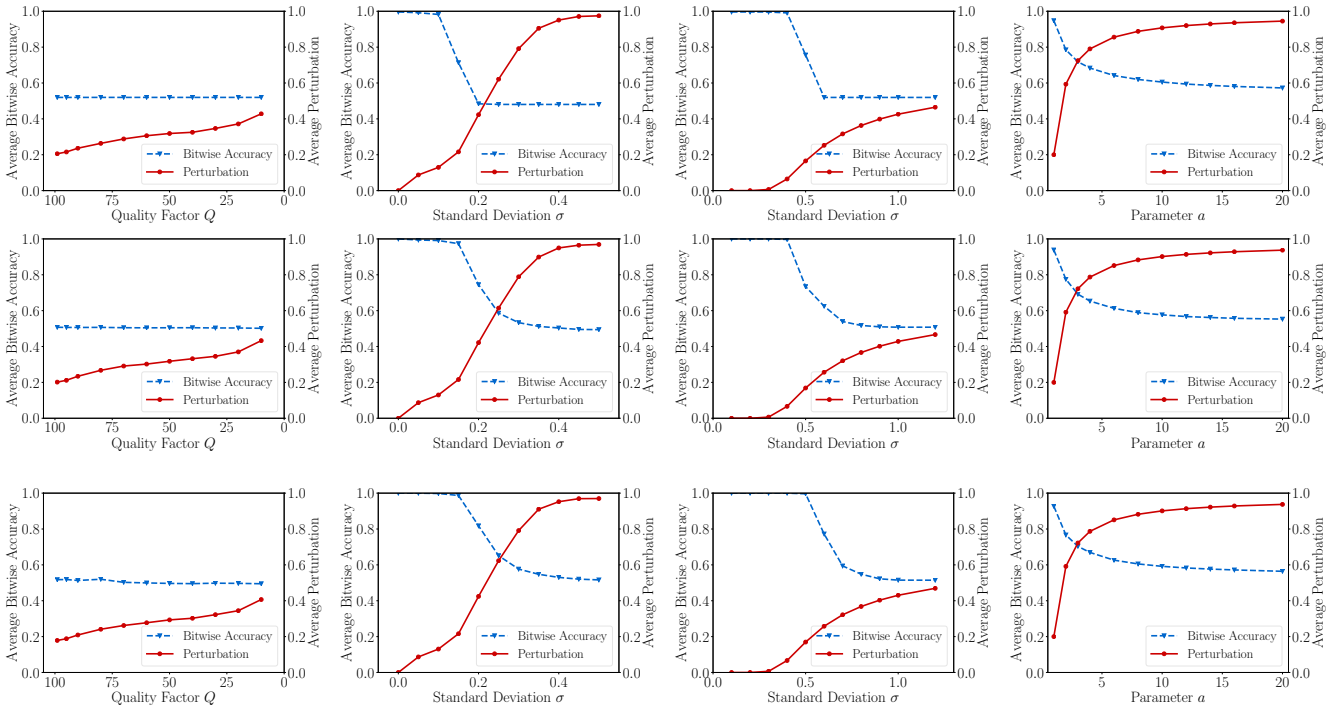
$$\begin{aligned}|w_t - w|_1 &= |w_t - w_{I_{pw}} + w_{I_{pw}} - w|_1 \\ &\leq |w_t - w_{I_{pw}}|_1 + |w_{I_{pw}} - w|_1 \leq \epsilon n + |w_{I_{pw}} - w|_1.\end{aligned}$$

Therefore, we have:

$$|w_t - w|_1 - \epsilon n \leq |w_{I_{pw}} - w|_1 \leq |w_t - w|_1 + \epsilon n.$$

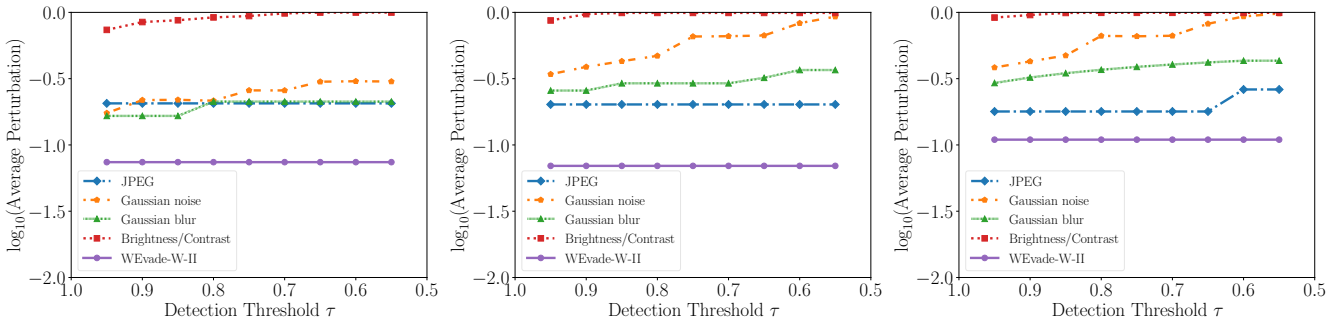
Thus, we have:

$$\begin{aligned}&\Pr(1 - \tau \leq BA(D(I_{pw}), w) \leq \tau) \\ &= \Pr(1 - \tau \leq BA(w_{I_{pw}}, w) \leq \tau) \\ &= \Pr(1 - \tau \leq 1 - \frac{|w_{I_{pw}} - w|_1}{n} \leq \tau)\end{aligned}$$



(a) JPEG (b) Gaussian noise (c) Gaussian blur (d) Brightness/Contrast

Figure 23: Average bitwise accuracy and average perturbation of the post-processed watermarked images when an existing post-processing method uses different parameter values. The watermarking method is UDH. The datasets are COCO (first row), ImageNet (second row), and CC (third row).



(a) COCO (b) ImageNet (c) CC

Figure 24: Average perturbation added by each post-processing method to evade the double-tail detector with different threshold τ in the white-box setting. We set the parameters of existing post-processing methods such that they achieve the same evasion rate as our WEvade-W-II. The watermarking method is UDH.

$$
\begin{aligned}
&= \Pr((1-\tau)n \le |w_{I_{pw}} - w|_1 \le \tau n) \\
&= 1 - \Pr((1-\tau)n > |w_{I_{pw}} - w|_1) - \Pr(|w_{I_{pw}} - w|_1 > \tau n) \\
&\ge 1 - \Pr((1-\tau)n > |w_t - w|_1 - \epsilon n) - \Pr(|w_t - w|_1 + \epsilon n > \tau n) \\
&= 1 - \Pr((1-\tau+\epsilon)n > |w_t - w|_1) - \Pr(|w_t - w|_1 > (\tau-\epsilon)n) \\
&= 1 - 2\Pr(|w_t - w|_1 > (\tau-\epsilon)n) \\
&= 1 - 2(1 - \Pr(|w_t - w|_1 \le (\tau-\epsilon)n)) \\
&= 2\Pr(|w_t - w|_1 \le (\tau-\epsilon)n) - 1.
\end{aligned}
$$

Since $|w_t - w|_1 \sim B(n, 0.5)$, we have:

$$
\begin{aligned}
&\Pr(1-\tau \le BA(D(I_{pw}), w) \le \tau) \\
&\ge 2\Pr(|w_t - w|_1 \le (\tau-\epsilon)n) - 1 \\
&= 2\Pr(|w_t - w|_1 \le \lfloor(\tau-\epsilon)n\rfloor) - 1 \\
&= 2P(\lfloor(\tau-\epsilon)n\rfloor) - 1.
\end{aligned}
$$

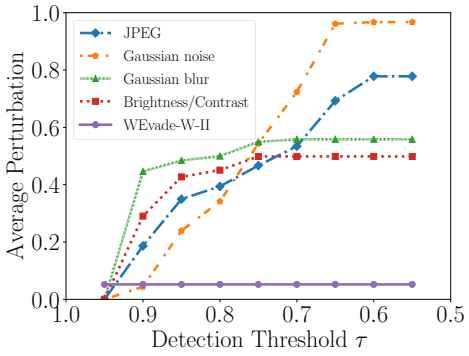
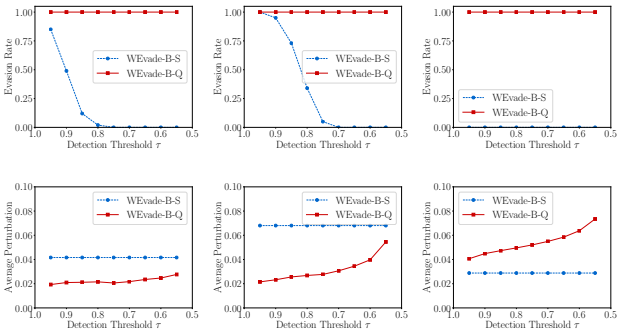
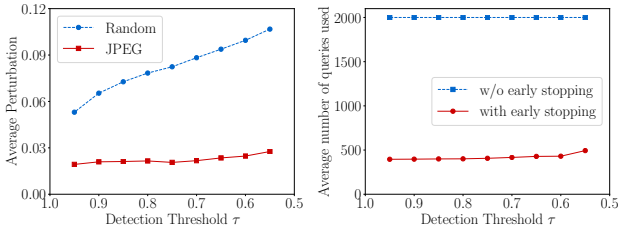


Figure 25: Average perturbation added by each post-processing method to evade the double-tail detector with different threshold τ for the COCO dataset. We set the parameters of existing post-processing methods such that they achieve the same evasion rate as WEvade-W-II. The watermarking method is HiDDeN and adversarial training is used. After adversarial training, the average bitwise accuracy is around 0.87. When τ is 0.95, empirical FNR is 99.6%, and thus existing post-processing methods do not add perturbations to a large fraction of watermarked images based on how we evaluate them, leading to 0 perturbations. However, they need much larger perturbations when τ is smaller than 0.9.



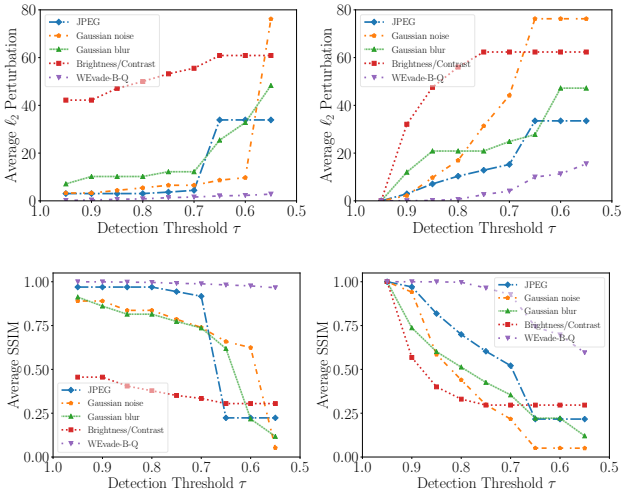
(a) COCO **(b) ImageNet** **(c) CC**

Figure 26: Comparing evasion rates (*first row*) and average perturbations (*second row*) of WEvade-B-S and WEvade-B-Q in the black-box setting. Watermarking method is UDH.



(a) Initialization **(b) Early stopping**

Figure 27: Impact of (a) initialization and (b) early stopping on our WEvade-B-Q for UDH and COCO dataset.



(a) Standard training **(b) Adversarial training**

Figure 28: Average perturbation, measured by ℓ_2-norm (first row) or SSIM (second row), added by each post-processing method to evade the double-tail detector with different τ in the black-box setting. WEvade-B-Q always achieves evasion rate 1, and we set the parameters of existing post-processing methods such that they achieve evasion rates as close to 1 as possible. The watermarking method is HiDDeN and dataset is COCO. When generating these perturbations, we change the ℓ_∞-norm to ℓ_2-norm at Line 16 in Algorithm 3.

Figure 29: DALL-E generated image with a visible watermark at the bottom right corner.

D PROOF OF THEOREM 4

For single-tail detector, we denote $D'(I_{pw}) = w'_{I_{pw}}$. According to Equation 9, we have:

$$BA(w'_{I_{pw}}, w_t) = 1 - \frac{|w'_{I_{pw}} - w_t|_1}{n} \leq 1 - \epsilon,$$
$$\implies |w'_{I_{pw}} - w_t|_1 \leq \epsilon n.$$

Then, according to the triangle inequality, we have:

$$|w'_{I_{pw}} - w|_1 = |w'_{I_{pw}} - w_t + w_t - w|_1$$
$$\leq |w'_{I_{pw}} - w_t|_1 + |w_t - w|_1 \leq \epsilon n + |w_t - w|_1.$$

Algorithm 3 WEvade-B-Q

Input: API of the target detector, a watermarked image I_w, query budget max_q, and early stop threshold ES.
Output: Post-processed image I_{pw}

$q \leftarrow 0$
//Initializing I_{pw}
for $Q \in [99, 90, 70, 50, 30, 10, 1]$ **do**
  $q \leftarrow q + 1$
  if $API(\text{JPEG}(I_w, Q))$=="non-AI-generated" **then**
    $I_{pw} \leftarrow \text{JPEG}(I_w, Q)$
    break
  end if
end for
//Iteratively move I_{pw} towards I_w
$\delta_{min} \leftarrow I_{pw} - I_w$
$es \leftarrow 0$
while $q \leq max\_q$ and $es \leq ES$ **do**
  $I_{pw}, q' \leftarrow \text{HopSkipJump}(I_{pw})$
  $q \leftarrow q + q'$
  if $\|I_{pw} - I_w\|_\infty < \|\delta_{min}\|_\infty$ **then**
    $\delta_{min} \leftarrow I_{pw} - I_w$
    $es \leftarrow 0$
  else
    $es \leftarrow es + 1$
  end if
end while
return $I_w + \delta_{min}$

Similarly, we have:

$$\begin{aligned}|w_t - w|_1 &= |w_t - w'_{I_{pw}} + w'_{I_{pw}} - w|_1 \\ &\le |w_t - w'_{I_{pw}}|_1 + |w'_{I_{pw}} - w|_1 \le \epsilon n + |w'_{I_{pw}} - w|_1.\end{aligned}$$

Therefore, we have:

$$|w_t - w|_1 - \epsilon n \le |w'_{I_{pw}} - w|_1 \le |w_t - w|_1 + \epsilon n.$$

Moreover, according to Definition 1, we have:

$$\begin{aligned}&\Pr(BA(w'_{I_{pw}}, w_{I_{pw}}) \ge \beta) \\ &= \Pr(1 - \frac{|w'_{I_{pw}} - w_{I_{pw}}|_1}{n} \ge \beta) \ge \gamma, \\ &\Longrightarrow \Pr(|w'_{I_{pw}} - w_{I_{pw}}|_1 \le (1-\beta)n) \ge \gamma.\end{aligned}$$

Thus, we have:

$$\begin{aligned}&\Pr(BA(D(I_{pw}), w) \le \tau) \\ &= \Pr(BA(w_{I_{pw}}, w) \le \tau) \\ &= \Pr(1 - \frac{|w_{I_{pw}} - w|_1}{n} \le \tau) \\ &= \Pr(|w_{I_{pw}} - w|_1 \ge (1-\tau)n).\end{aligned}$$

Then, according to the triangle inequality, we have:

$$\begin{aligned}&|w'_{I_{pw}} - w|_1 = |w'_{I_{pw}} - w_{I_{pw}} + w_{I_{pw}} - w|_1 \\ &\le |w'_{I_{pw}} - w_{I_{pw}}|_1 + |w_{I_{pw}} - w|_1, \\ &\Longrightarrow |w_{I_{pw}} - w|_1 \ge |w'_{I_{pw}} - w|_1 - |w'_{I_{pw}} - w_{I_{pw}}|_1.\end{aligned}$$

Similarly, we have:

$$\begin{aligned}|w_{I_{pw}} - w|_1 &= |w_{I_{pw}} - w'_{I_{pw}} + w'_{I_{pw}} - w|_1 \\ &\le |w'_{I_{pw}} - w|_1 + |w_{I_{pw}} - w'_{I_{pw}}|_1.\end{aligned}$$

Thus, we have:

$$\begin{aligned}&\Pr(|w_{I_{pw}} - w|_1 \ge (1-\tau)n) \\ &\ge \Pr(|w'_{I_{pw}} - w|_1 - |w'_{I_{pw}} - w_{I_{pw}}|_1 \ge (1-\tau)n) \\ &\ge \Pr(|w'_{I_{pw}} - w|_1 - (1-\beta)n) \ge (1-\tau)n) \\ &\quad \cdot \Pr(|w'_{I_{pw}} - w_{I_{pw}}|_1 \le (1-\beta)n) \\ &\ge \Pr(|w'_{I_{pw}} - w|_1 \ge (2-\tau-\beta)n) \cdot \gamma \\ &\ge \gamma \Pr(|w_t - w|_1 - \epsilon n \ge (2-\tau-\beta)n) \\ &= \gamma \Pr(|w_t - w|_1 \ge (2-\tau-\beta+\epsilon)n).\end{aligned}$$

Since $|w_t - w|_1 \sim B(n, 0.5)$, we have:

$$\begin{aligned}&\Pr(BA(D(I_{pw}), w) \le \tau) \\ &\ge \gamma \Pr(|w_t - w|_1 \ge \lceil (2-\tau-\beta+\epsilon)n \rceil) \\ &= \gamma(1 - P(\lceil (2-\tau-\beta+\epsilon) \rceil)) \\ &= \gamma P(\lfloor (\tau+\beta-\epsilon-1)n \rfloor).\end{aligned}$$

For double-tail detector, we have:

$$\begin{aligned}&\Pr(1-\tau \le BA(D(I_{pw}), w) \le \tau) \\ &= \Pr(1-\tau \le BA(w_{I_{pw}}, w) \le \tau) \\ &= \Pr(1-\tau \le 1 - \frac{|w_{I_{pw}} - w|_1}{n} \le \tau) \\ &= \Pr((1-\tau)n \le |w_{I_{pw}} - w|_1 \le \tau n) \\ &= 1 - \Pr((1-\tau)n > |w_{I_{pw}} - w|_1) - \Pr(|w_{I_{pw}} - w|_1 > \tau n) \\ &\ge 1 - \Pr(|w'_{I_{pw}} - w|_1 - |w'_{I_{pw}} - w_{I_{pw}}|_1 < (1-\tau)n) \\ &\quad - \Pr(|w'_{I_{pw}} - w|_1 + |w_{I_{pw}} - w'_{I_{pw}}|_1 > \tau n) \\ &\ge \Pr(|w'_{I_{pw}} - w|_1 - |w'_{I_{pw}} - w_{I_{pw}}|_1 \ge (1-\tau)n) \\ &\quad + \Pr(|w'_{I_{pw}} - w|_1 + |w_{I_{pw}} - w'_{I_{pw}}|_1 \le \tau n) - 1 \\ &\ge \gamma P(\lfloor (\tau+\beta-\epsilon-1)n \rfloor) + \Pr(|w'_{I_{pw}} - w|_1 + (1-\beta)n \le \tau n) \\ &\quad \cdot \Pr(|w'_{I_{pw}} - w_{I_{pw}}|_1 \le (1-\beta)n) - 1 \\ &\ge \gamma P(\lfloor (\tau+\beta-\epsilon-1)n \rfloor) \\ &\quad + \Pr(|w_t - w|_1 + \epsilon n \le (\tau+\beta-1)n) \cdot \gamma - 1 \\ &\ge \gamma P(\lfloor (\tau+\beta-\epsilon-1)n \rfloor) \\ &\quad + \gamma \Pr(|w_t - w|_1 \le (\tau+\beta-\epsilon-1)n) - 1 \\ &\ge 2\gamma P(\lfloor (\tau+\beta-\epsilon-1)n \rfloor) - 1.\end{aligned}$$