WRATH: Turning Watermark Robustness Against Itself via a Watermark-Agnostic Black-Box Invalidation Attack

Nan Jiang*, Juan Hu*, Bangjie Sun*, Terence Sim*, and Jun Han†
*National University of Singapore, Singapore
†KAIST, South Korea

***Abstract*—Watermarks are increasingly embedded in AI-generated images to indicate their machine-generated origin. For reliable detection, watermarks are designed to remain robust against common image manipulations such as resizing and compression. In this paper, we uncover a previously overlooked vulnerability: the robustness property of watermarking schemes can inadvertently expose the image features that carry watermark signals. Exploiting this vulnerability, we introduce *WRATH*, the first watermark-agnostic black-box attack capable of both watermark removal and forgery. Our attack is practical and requires only a scheme's robustness information, which is typically public or can be obtained through simple robustness tests. When evaluated against state-of-the-art watermarking schemes, including Amazon's, *WRATH* successfully attacks all evaluated schemes while preserving high perceptual image quality. We also discuss practical defenses to mitigate this vulnerability, and our findings call for a rethink of watermark security.**

1. Introduction

As generative AI continues to advance, its ability to produce realistic-looking images has raised growing concerns about misinformation [1], [2]. In response, governments have begun mandating safeguards for generative AI. For example, California has enacted legislation requiring major developers to provide tools for detecting AI-generated content, with the law set to take effect in 2026 [3]. Reflecting this regulatory direction, watermarking is increasingly being adopted as a standard practice across the industry. Several major providers have already implemented such solutions – for instance, Amazon's Titan and Google's Imagen embed invisible watermarks in their generated images [4], [5], and Amazon offers a watermarking detection API, enabling the general public to verify if an image contains Amazon's watermark and thus is generated by Amazon [4].

This regulatory push has also reignited interest in digital watermarking research. Much of the recent work focuses on improving watermark robustness against ***removal attacks*** [6], [7], [8], [9], [10], [11], where a watermarked image is modified, for example through operations such as compression, cropping, or adversarial perturbations, to render the embedded watermark undetectable. A robust watermark should remain detectable even when the image undergoes such transformations, whether introduced intentionally by an attacker or unintentionally through common image processing. In addition to removal robustness, some works also aim to improve watermark robustness against ***forgery attacks*** [12], [13], in which a non-watermarked image is processed to deceive the watermark detector into falsely identifying it as watermarked. Robustness against such attacks is crucial to ensure that AI providers are not falsely attributed as the source of images that were not generated by their systems.

Despite the extensive effort devoted to improving watermark robustness, we show that robustness itself can be turned against the watermark. Specifically, an attacker can observe the robustness characteristics of a watermarking scheme. By tracking which manipulations (e.g., resizing, cropping, compression) preserve or remove the watermark, they can infer the specific image features containing the signal. We refer to these features as ***watermark carriers***. We further refer to manipulations that remove the watermark as ***fragile manipulations***, and those that preserve it as ***robust manipulations***. Under this terminology, watermark signals are likely embedded in image features that remain stable under robust manipulations yet are disrupted by fragile ones. We term this vulnerability the **W**atermark **R**obustness **A**gainst **T**he **H**ost Image (*WRATH*). Importantly, *WRATH* arises directly from the observable robustness characteristics of watermarking schemes and therefore does not depend on how the watermark is embedded, making *WRATH* largely watermark agnostic. Furthermore, we observe that many state-of-the-art watermarking schemes embed watermark signals into consistent carriers across images. While not every watermarking design follows this pattern, our experiments show that it holds across a broad and representative set of schemes. This consistency enables attackers to exploit *WRATH* by aggregating information across multiple images, facilitating the identification of watermark carriers. Once watermark carriers are identified, an attacker can (i) disrupt them in a watermarked image to remove the watermark, or (ii) modify them in a non-watermarked image to forge watermark signals.

Exploiting *WRATH*, we develop the *practical*, *query-based WRATH* attack as a proof-of-concept exploitation of the identified vulnerability. *WRATH* attack achieves two forms of generality. First, it is *watermark-agnostic*, as it relies only on observable robustness characteristics and does

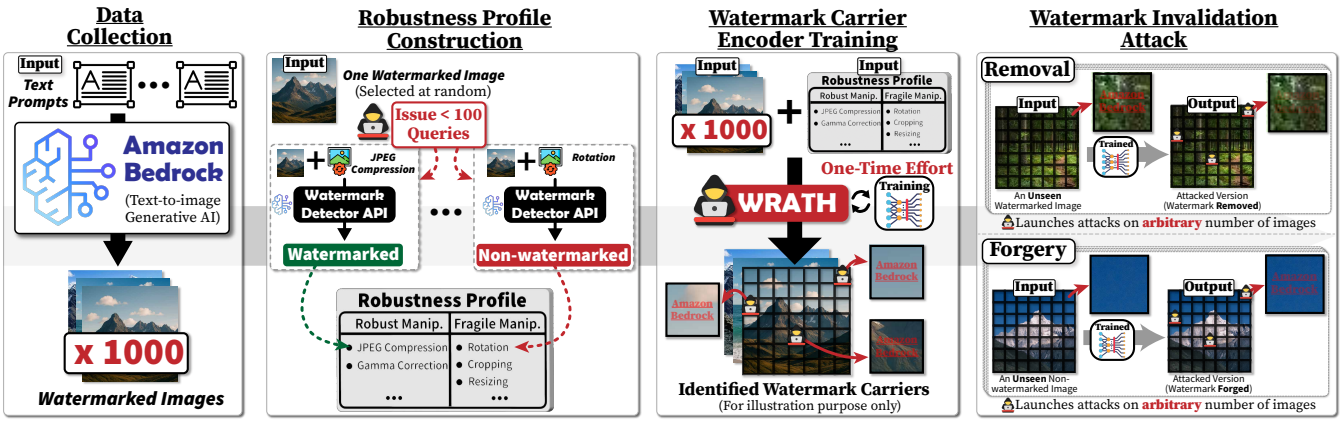


Figure 1. Figure depicts the high-level concept of our attack targeting Amazon's watermark. **Data collection**. An adversary generates multiple watermarked images using Amazon Titan's API. **Robustness Profile Construction**. The adversary queries the public watermark detector API with *only one* watermarked image processed by different image manipulations (e.g., rotation, JPEG compression) and records the binary responses ("watermarked" or "non-watermarked"). These responses form a *robustness profile* that indicates the robust and fragile manipulations. **Watermark Carrier Encoder Training**. The adversary uses the robustness profile (i.e., constructed from a single image) to train a deep learning model (i.e., encoder) on the collected dataset. The resulting encoder identifies watermark carriers and the embedded signal patterns. This training is a one-time effort. **Watermark Invalidation Attack**. The adversary launches attacks using the trained encoder on an arbitrary number of images by either disrupting the inferred carriers in any watermarked image to remove the watermark, or injecting similar signal patterns into carriers of a non-watermarked image to forge a watermark. We recommend viewing this figure in color.

not require knowledge of the watermarking algorithm. Second, it supports both watermark *removal* and watermark *forgery*, which we collectively refer to as watermark ***invalidation attacks***. To the best of our knowledge, this is the first unified attack pipeline that simultaneously enables removal and forgery while generalizing across different watermarking schemes. Fig. 1 illustrates the high-level workflow of our attack using Amazon's watermarking system as an example. The adversary first generates watermarked images using Amazon Titan's API and queries the public watermark detection API with a single watermarked image under different image manipulations (e.g., resizing, cropping, JPEG compression). The resulting binary responses form a ***robustness profile*** that distinguishes robust and fragile manipulations. Using this profile, the adversary trains a deep learning encoder to identify watermark carriers and the associated watermark signal patterns. Once trained, the encoder can be used to guide attacks that either disrupt the identified carriers in watermarked images to remove the watermark or inject similar signal patterns into carriers of non-watermarked images to forge a watermark.

The *WRATH* attack is practical as it operates in a *black-box* setting and requires only a realistic query budget. The attack relies solely on generative model outputs to collect watermarked images and on detector outputs to construct the robustness profile, with no access to either system's internals. We assume access to a detector API that returns only a binary label (i.e., "watermarked" or "non-watermarked"), representing a minimal-feedback setting. We build the robustness profile by issuing carefully chosen queries to the detector. In our experiment, *WRATH* works under realistic query budgets (i.e., < 100 queries). We infer the watermark carriers by training an encoder over 1,000 images produced by the target image generative model. Generating this dataset on Amazon Titan costs approximately US$14.40. This process is a one-time effort per scheme, and the resulting encoder can be used to attack an arbitrary number of images.

We evaluate *WRATH* on seven state-of-the-art watermarking schemes, including a deployed system (Amazon Titan) and six academic methods [6], [7], [10], [11], [12], [14] that span a broad spectrum of existing watermark designs. Our experiments reveal that all evaluated schemes are susceptible to *WRATH*. Using identified watermark carriers, we can differentiate watermarked and non-watermarked counterparts with up to 93.57% accuracy. In addition, experiments show that *WRATH*'s watermark removal performance is comparable to tailored removal attacks. Across the evaluated schemes, our attack is able to remove watermark while preserving perceptual quality, achieving removal success rates up to 95%. Likewise, we can forge all evaluated schemes with high visual fidelity, reaching forgery success rates up to 100%. Furthermore, we analyze the root cause of *WRATH* and discuss potential defenses. Our findings challenge conventional watermark design goals, emphasize the need to reevaluate the security of watermarking schemes, and provide insights into their future development. We summarize our contributions as follows:

(1) **We identify a previously overlooked vulnerability**. We show that robustness profiles reveal watermark carriers. The common practice of embedding watermark signals into consistent carriers enables aggregation of information across images to learn both carrier and watermark signal patterns.

(2) **We introduce a practical watermark invalidation attack**. Leveraging *WRATH*, our attack achieves both removal and forgery under realistic attack assumptions and generalizes across various watermarking schemes.

(3) **Our experiments demonstrate that many state-of-the-art schemes are vulnerable to *WRATH*.** Using identified watermark carriers, we can differentiate watermarked and non-watermarked counterparts with up to 93.57% accuracy.
(4) **We demonstrate end-to-end invalidation effectiveness against seven leading watermarking schemes, including Amazon's, while preserving high image quality**.
(5) **We analyze potential countermeasures and provide design guidelines to safeguard watermarking schemes against our attack**.

2. Background and Related Work

2.1. Image Watermarking

Categorization of Existing Watermarking Schemes. Existing watermarking schemes can be categorized as either ***non-semantic*** [6], [8], [12], [14], [15] or ***semantic*** [10], [11], [13], [16], [17]. Non-semantic watermarks embed signals into low-level image features (e.g., pixel or frequency components) without affecting the image's semantics (e.g., the objects or scenes the image portrays). Semantic watermarks embed signals into the initial noise of the diffusion process, thereby influencing the image generation trajectory and encoding the watermark within the semantic content of the generated image. Equivalently, semantic schemes introduce substantial modifications to the image and are considered ***high-perturbation***,[1] while many non-semantic schemes introduce only imperceptible changes to the image and are therefore ***low-perturbation***. Notably, some non-semantic methods (e.g., StegaStamp [7]) still qualify as high-perturbation due to the visible artifacts they introduce. Prior work reports that high-perturbation watermarks are typically embedded in the image's low-frequency components, whereas low-perturbation watermarked are typically embedded to high-frequency components [18].

Watermark Detection. In typical image watermarking, a *binary string* (e.g., `001101`) is embedded into the image to encode metadata, such as the image's source. This is the most common type of embedded information, although other forms, such as images, could also be used. To determine whether an image is watermarked, the detector first decodes a binary string from the image, then computes the fraction of bits that match the expected watermark code (i.e., ***bit accuracy***). If the bit accuracy exceeds a predefined threshold, the image is considered "watermarked". Typically, the detector only outputs categorical responses rather than detailed bit accuracy. For example, Amazon's detector outputs either "No Watermark Detected" or "Watermark Detected" with "Low", "Medium", or "High" confidence.

1. High-perturbation does not necessarily imply visible watermarking, where a visible tag is attached to the image; it also includes in-model schemes that influence the process of image generation and thereby affect the resulting content. In this paper, we define high-perturbation to exclude visible watermarks.

2.2. Attacks against Image Watermarking

In this section we survey prior attacks against image watermarking. Tab. 1 summarizes the literature by *attack scope* (i.e., removal, forgery, or full invalidation), *targeted scheme families* (i.e., semantic or non-semantic) and its *threat-model practicality*. We first define the terminology used in the table, then review representative attacks and their limitations, and conclude by positioning our contributions relative to prior work.

Attack scope. "Removal only (R)" denotes attacks that are *only* capable of removing a watermark – i.e., make a watermarked image be recognized as non-watermarked. "Forgery only (F)" denote attacks that are *only* capable of forging a watermark – i.e., make a clean image be recognized as watermarked. "Invalidation (R+F)" denotes a *single* attack pipeline that is capable of both removing and forging watermarks. **Targeted scheme family.** "Non-semantic only" denotes attacks that are effective or evaluated only against non-semantic watermarks. "Semantic only" denotes attacks that are effective or evaluated only against semantic watermarks. "Both" denotes attacks shown to work on both families. **Threat-model practicality**. "Practical" denotes realistic, deployable attacker assumptions: black-box access to a detector API that returns only binary responses ("watermarked" and "non-watermarked"), no knowledge on the watermarking scheme (i.e., watermark type, model weights and training data), no access to non-watermarked images or surrogate models. "Not practical" denotes stronger capabilities that go beyond these assumptions.

We survey 25 recent attacks and summarize each work's attack scope, targeted scheme family, and threat-model practicality in Tab. 1. We observe that roughly half of the surveyed attacks (13 out of 25) focus exclusively on watermark removal, three of those operate under practical threat models and generalize across scheme families. We attribute the popularity of removal-only attacks to operational factors. Removal attacks typically only require disrupting whatever image features that *potentially* carry the watermark signal, so it needs less precise information about where the actual carriers are located, making practical removal attacks easy to execute. For example, some practical attacks randomly disrupt image features in the hope that watermarks will be removed as a side effect. A typical example of this is regeneration-based attacks [22], [30], where a watermarked image is passed through a generative model to produce a potentially non-watermarked version. A more advanced practical removal attack relies on an assumption that the watermark signal resides in the image's frequency features [18]. However, it merely determines whether watermarks reside in low or high-frequency bands, limiting its effectiveness in thoroughly removing watermarks across various schemes when image content must be preserved (i.e., no cropping). As shown in §5.3, its success typically requires destructive edits such as cropping. Compared to removal attacks, forgery-capable attacks (i.e., forgery-only and invalidation) mainly target non-semantic watermarks. To the best of our knowledge, only one prior attack generalizes forgery across

TABLE 1. CAPABILITY MATRIX OF ATTACKS ON IMAGE WATERMARKING.

Attack scope \ Scheme family	Non-semantic only		Semantic only		Both	
	Not practical	**Practical**	**Not practical**	**Practical**	**Not practical**	**Practical**
Removal only (R)	[19], [20], [21]	[22], [23], [24]	[25]	[26]	[27], [28]	[18], [29], [30]
Forgery only (F)	[31], [32]	[33], [34]	[35]	-	-	[28]
Invalidation (R + F)	[36]	[37], [38], [39]	[40]	[41], [42]	-	**Ours**

Rows denote the **attack scope**: "Removal only (R)" denotes attacks that are *only* capable of removing a watermark. "Forgery only (F)" denote attacks that are *only* capable of forging a watermark. "Invalidation (R+F)" denotes a *single* attack pipeline that is capable of removing and forging watermarks. Columns denote the **targeted scheme family**: "Non-semantic only" denotes attacks that are effective or evaluated only against non-semantic watermarks. "Semantic only" denotes attacks that are effective or evaluated only against semantic watermarks. "Both" denotes attacks shown to work on both families. Each main column is split into two sub-columns indicating **threat-model practicality**. "Practical" denotes realistic, deployable attacker assumptions: black-box access to a detector API that returns only binary responses ("watermarked" and "non-watermarked"), no knowledge on the watermarking scheme (i.e., watermark type, model weights and training data), and no access to non-watermarked images or surrogate models. "Not practical" denotes stronger capabilities that go beyond these assumptions. A citation in a cell lists a representative work whose claimed capability and evaluated threat model match that row and sub-column. We record each paper's stated capabilities and threat model as reported in the original work to avoid overstating.

different watermarking scheme families [28].[2] We attribute the imbalance between removal and forgery research to practical and technical challenges. Forgery requires reproducing the *exact* watermark signal in a non-watermarked image, which is inherently more difficult than removal attacks. To our knowledge, no prior invalidation attack operates under a realistic threat model and generalizes across watermarking scheme families.

We fill this gap by proposing a *watermark-agnostic*, *practical*, *query-based invalidation* attack. Our attack exposes a previously overlooked vulnerability that by observing a watermarking scheme's robustness characteristic – that is, which image manipulations (e.g., resizing, cropping, compression) preserve the watermark and which remove it – we can infer the image features that carry the watermark signal, denoted as watermark carrier. The vulnerability is amplified by a common design choice: many state-of-the-art watermarking schemes embed signals into consistent carriers across images, enabling an adversary to aggregate observations and reliably learn both the carriers and the embedded watermark signal patterns.

3. Threat Model

We consider two parties: the *image generative model owner* and the *adversary*. The model owner runs a proprietary generative model and charges users a subscription fee to prompt it (e.g., through an API). Upon receiving a prompt, the model owner chooses a watermarking scheme (semantic or non-semantic) based on factors such as robustness, visibility, and computational cost, and then embeds the watermark into the generated image to indicate its origin. We assume that a single watermarking scheme is used across all generated images within a deployment period, with periodic updates when necessary. Beyond the image generator, the model owner runs a proprietary watermark detector (e.g., via an API) that allows the public to verify whether an image is generated by the generator.

2. This related work proposes both removal and forgery attacks for two watermark scheme families, but designs them separately rather than as a unified pipeline. Therefore, we do not classify it as an invalidation attack.

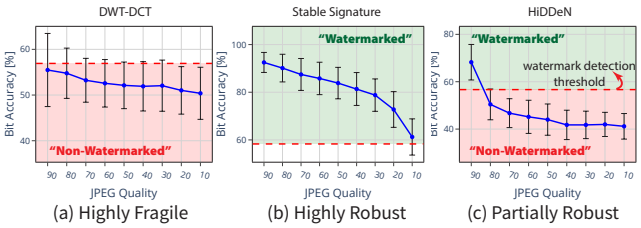


Figure 2. Robustness behaviors of watermarking schemes under JPEG compression. Each curve shows watermark bit accuracy while decreasing JPEG quality. (a) DWT–DCT is highly fragile, failing even at the highest quality. (b) Stable Signature is highly robust under JPEG compression, maintaining detectability even at quality = 10; however, the watermark signals are nearly destroyed at this setting, with bit accuracy approaching the detection threshold. (c) HiDDeN exhibits partial robustness. **We report bit accuracy only to illustrate that watermark removal progresses gradually as manipulation intensity increases. *WRATH* does not use bit accuracy from the detector at any stage.**

Adversary's Goals. The adversary aims to launch watermark invalidation attack, encompassing both watermark removal and forgery, to deceive the watermark detector. In the removal attack, the adversary's goal is to modify any image generated by the generative model so that the proprietary watermark detector incorrectly recognize it as not containing the model's watermark. In the forgery attack, the goal is to modify any non-watermarked image so that the proprietary watermark detector incorrectly recognize it as containing the model's watermark. In both attacks, the adversary aims to apply only small perturbations so that modified images remain perceptually similar to the original.

Adversary's Capabilities. Since the target watermark detector and image generative model are proprietary, the adversary has only *black-box* access (i.e., API access): they can query them but have no knowledge of their structure and parameters. Additionally, we assume that the adversary operates under *reasonable* financial and computational constraints. Concretely, monetary costs associated with image generation restrict large-scale collection of watermarked images, and limited compute time and potential provider usage policies restrict extensive probing of the watermark detector.

We assume that the watermark detector returns only a binary response (i.e., "watermarked" or "non-watermarked"). Consequently, the adversary can build the robustness profile only from the detector's binary responses.

4. Design of *WRATH* Attack

In this section, we present a proof-of-concept exploitation of the *WRATH* vulnerability. Our design is motivated by a common design choice in many existing watermarking schemes: watermark signals are embedded into consistent carriers across generated images. This consistency allows us to aggregate information across images using data-driven methods to identify potential watermark carriers.[3]

4.1. Overview

The design of *WRATH* attack is divided into four steps as depicted in Fig. 1. **Dataset Collection**. The first step of the attack is to collect a dataset of images watermarked by the targeted proprietary image generator. Since the adversary has query access to the target model, they can simply query it to generate the watermarked images. **Robustness Profile Construction** (§4.2). Then, the adversary selects *one* watermarked image at random from the collected dataset and selects a set of common image manipulations (e.g., cropping, resizing, JPEG compression). For each manipulation, they apply it to the selected image independently, query the watermark detector with the processed image, and record the detector's binary response ("watermarked" or "non-watermarked"). The collection of these responses constitutes a robustness profile that indicates which manipulations remove the watermark (i.e., the detector returns "non-watermarked") and which preserve it (i.e., the detector returns "watermarked") for the selected image. We define manipulations that remove the watermark as *fragile manipulations* and those preserve the watermark as *robust manipulations*. **Watermark Carrier Encoder Training** (§4.3). Next, using the dataset of watermarked images and the robustness profile, the adversary trains a watermark carrier encoder. The encoder is trained to (i) localize watermark carriers in an input image, namely the image features that carry the watermark signal, and (ii) encodes the watermark carriers into a compact feature embedding, hereafter referred to as the ***watermark embedding***. The training leverages the insight that watermark signals are likely embedded in image features that remain stable under robust manipulations yet are disturbed by fragile ones. **Watermark Invalidation Attack** (§4.4). Finally, the adversary uses the trained watermark-carrier encoder to mount (i) watermark-removal (§4.4.1) and (ii) watermark-forgery (§4.4.2) attacks against the target image generator. For removal, given a previously unseen image produced by the generator, i.e., one not used during training, the adversary applies modifications to the image that push its watermark embedding away from its original value. For forgery, given a non-watermarked image, the adversary applies modifications to the image that drive its watermark embedding toward that of images produced by the target generator.

3. We also discuss the feasibility of a non-data-driven approach that does not exploit the consistent-carrier property in §B.1.

4.2. Robustness Profile Construction

This module constructs a robustness profile that records which image manipulations are robust and which are fragile. We label a manipulation by applying it to a *single* watermarked image and querying the watermark detector on the resulting image. If the detector outputs "watermarked", we label it robust, otherwise fragile. However, the space of image manipulations is vast, so exhaustively enumerating and labeling all manipulations is not viable for a computationally constrained adversary. To address this, we design a *query-efficient* algorithm that *adaptively* samples manipulations from the vast manipulation space and labels them based on the detector's responses. The resulting profile provides useful cues that we later use to identify the watermark carriers (§4.3). We begin by explaining the intuition behind the algorithm and then present the detailed procedure.

Intuition. An image manipulation is defined by two components: a *type*, which specifies the operation, and a *parameter*, which controls its intensity. For example, in the manipulation "rotate an image by 10 degrees", *rotation* is the manipulation type, and *10 degrees* is the parameter that determines how much the image is rotated. Many manipulation types have continuous parameter spaces, making the set of manipulations infinite. In contrast, the number of manipulation types is finite and can be enumerated. We therefore adopt a per-type strategy: given a manipulation type, we adaptively sample its parameter over a specified range and label the resulting manipulation as robust or fragile. We design our algorithm based on two insights.

Insight ❶ The robustness profile reveals where watermark carriers are likely to reside: fragile manipulations that remove the watermark tend to perturb watermark carriers, whereas robust manipulations that preserve the watermark tend to perturb non-carrier features. *We identify a feature as a carrier if it is disturbed by a fragile manipulation yet remains stable under a robust manipulation.* As manipulation intensity increases, more image features are disrupted. For fragile manipulations, increasing the intensity beyond the point that removes the watermark begins to disturb non-carrier features as well. Reducing the intensity of a robust manipulation causes more non-carrier features to remain stable. For precise localization, ***we thus favor fragile manipulations at the lowest intensity that removes the watermark, and robust manipulations at a higher intensity that preserves the watermark, in order to suppress as many non-carrier features as possible***.

Insight ❷ For a given manipulation type, a watermarking scheme exhibits one of three behaviors: ***highly robust*** (no intensity within the specified range removes the watermark), ***highly fragile*** (even the lowest intensity removes the watermark), or ***partially robust*** (some intensities remove the watermark). Fig. 2 illustrates how three watermarking schemes

behave differently under JPEG compression. Accordingly, for manipulation types under which the scheme is *highly fragile*, we label the manipulation type at its lowest intensity as fragile. Because the degree of watermark removal changes approximately monotonically with the manipulation intensity, for manipulation types under which the scheme is *partially robust*, we run a binary search over the parameter range to find the parameter that just removes the watermark, and label the corresponding manipulation as fragile. For manipulation types under which the scheme is *highly robust*, we label the manipulation type at its *medium* intensity as robust. Although higher-intensity robust manipulations can further suppress non-carrier features, they also risk disrupting a large fraction of carriers, as illustrated in Fig. 2c. The bit accuracy remains only marginally above the detection threshold, indicating that watermark carriers are nearly disrupted even though the detector still returns "watermarked". As a result, the detector's binary response provides no indication of whether a highest-intensity robust manipulation has already disrupted a large fraction of carriers. We therefore label the manipulation type with medium intensity as robust to balance localization specificity with carrier preservation.

Algorithm. The algorithm takes a single watermarked image, I_w, and a set of image manipulation types, $\mathbb{T} = \{t_1, t_2, \ldots, t_n\}$, and their corresponding parameter range, $\{[\theta_{t_i}^{low}, \theta_{t_i}^{high}] | t_i \in \mathbb{T}\}$, as input, and outputs a robustness profile that record the sets of *robust*, $\mathbb{M}_{robust}$, and *fragile* manipulations, $\mathbb{M}_{fragile}$. For each $t_i \in \mathbb{T}$, the algorithm samples parameter from $[\theta_{t_i}^{low}, \theta_{t_i}^{high}]$ and labels corresponding manipulation as robust or fragile following three steps.

Step ❶: query the watermark detector, $\mathcal{D}_w$, using I_w processed by the manipulation at its *lowest intensity*. If $\mathcal{D}_w$ returns "non-watermarked", the manipulation type at its lowest intensity is added to the fragile set.

Step ❷: otherwise, query $\mathcal{D}_w$ using I_w processed by the manipulation at its *highest intensity*. If $\mathcal{D}_w$ returns "watermarked", we include the manipulation type at *medium intensity* in the robust set.

Step ❸: otherwise, use binary search over $[\theta_{t_i}^{low}, \theta_{t_i}^{high}]$ to find the intensity value that just makes $\mathcal{D}_w$ output "non-watermarked", and add this manipulation to the fragile set.

4.3. Watermark Carrier Encoder Training

This module trains a watermark carrier encoder, $\mathcal{E}_{carrier}(\cdot)$, for a targeted watermarking scheme. The encoder takes an image as input, which may be represented either in its spatial domain or after a transformation into another domain (e.g., frequency or latent). Let $\mathcal{T} : \mathcal{X} \to \hat{\mathcal{X}}$ denote the image-domain transformation function. We write the encoder input as $\hat{I} = \mathcal{T}(I)$ for generality, with $\mathcal{T}$ being the identity function when operating in the spatial domain. The carrier encoder localizes watermark carriers in $\hat{I}$ and encodes them into a compact feature embedding $e_w \in \mathbb{R}^d$, which we refer to as the ***watermark embedding***. Our training strategy builds on the key insight that watermark carriers are likely to be the image features that remain stable under robust manipulations but are disrupted by fragile manipulations. Accordingly, we guide the carrier encoder to locate image features that remain similar between the original image and its robustly manipulated version (i.e., the same image processed by a robust manipulation), but become different between the original image and its fragilely manipulated version. We next describe our training strategy in detail.

Model Training. We train the watermark carrier encoder, $\mathcal{E}_{carrier}(\cdot)$, on a dataset of watermarked images from the target watermarking scheme, $\{\hat{I}_w^{(i)}\}_{i=1}^N$, all represented in a single, fixed domain (e.g., spatial, frequency, or latent). We employ two training strategies.

❶ ***Optimization using Triplet Loss.*** First, we employ *triplet loss* [43] as our training objective. $\mathcal{E}_{carrier}(\cdot)$ is trained on triplets consisting of: an *anchor*, which is the unprocessed watermarked image, $\hat{I}_w$; a *positive*, obtained by applying a robust manipulation to $\hat{I}_w$; and a *negative*, obtained by applying a fragile manipulation to $\hat{I}_w$. The triplet loss encourages the model to pull the *anchor* and the *positive* closer by learning features present in both, which are features that remain stable under robust manipulations. At the same time, it pushes the *anchor* and the *negative* apart by extracting features that differ between them, which are features sensitive to fragile manipulations. We refer to the features revealed by a single triplet as the ***candidate carrier set***. For each training triplet, manipulations are randomly sampled from the robust and fragile manipulation sets and applied accordingly. To improve localization, we aggregate *candidate sets* by training $\mathcal{E}_{carrier}(\cdot)$ on triplets constructed across all *anchor* images in $\{\hat{I}_w^{(i)}\}_{i=1}^N$. For each *anchor*, we construct multiple triplets, each with a distinct robust–fragile manipulation combination.

(2) Curriculum Learning. We observe significant variability in optimization difficulty across triplets. This variation is closely linked to the size of the *candidate carrier set* revealed by a particular triplet, as discussed in Appendix B.2. Triplets that lead to a larger candidate carrier set tend to be easier to optimize because they expose more informative features for the carrier encoder to learn from, whereas smaller sets provide limited feature information and are generally more difficult to optimize. Consequently, the model may become biased toward triplets with larger candidate sets, which are more likely to include non-carrier features. This bias makes precise carrier localization challenging. To address this issue, we adopt a curriculum training strategy [44]. Curriculum training helps by gradually increasing the difficulty of the training samples presented to the model. We begin training with easier triplets (i.e., those associated with larger candidate set), allowing the model to first identify the approximate location where the carrier may reside. As training progresses, we introduce more challenging triplets, encouraging the model to learn more precise carrier locations. This staged learning process mitigates the bias toward easier triplets and enhances the model's ability to identify candidate watermark carriers more precisely.

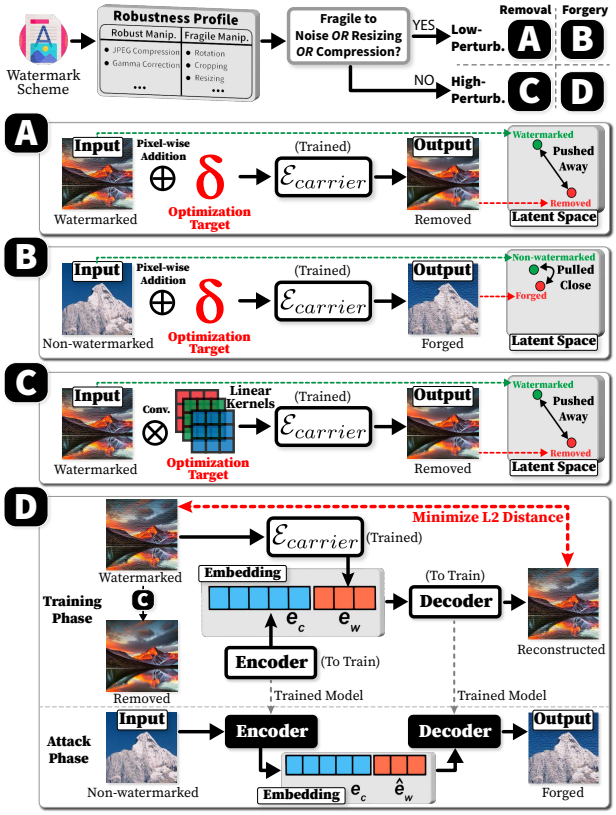


Figure 3. Overview of the invalidation attack pipeline. We obtain a scheme's robustness profile and categorize it as *low-* or *high-perturbation*, then select the corresponding attack: (A) removal – low-perturbation; (B) forgery – low-perturbation; (C) removal – high-perturbation; (D) forgery – high-perturbation. We recommend viewing this figure in color.

4.4. Watermark Invalidation Attack

We now present the design of our watermark invalidation pipeline. This pipeline supports both watermark removal (§4.4.1) and watermark forgery (§4.4.2), and leverages the trained watermark carrier encoder (§4.3). The workflow of our unified invalidation pipeline is summarized in §4.4.3.

4.4.1. Watermark Removal Attack. Given any watermarked image, I_w, and a trained *watermark carrier encoder*, $\mathcal{E}_{carrier}(\cdot)$, the watermark removal attack aims to generate an adversarial image, I_{adv}, that is visually similar to I_w yet is classified by the watermark detector as non-watermarked. As such, we introduce an optimization strategy to find the optimal I_{adv} under two objectives, namely *watermark carrier disruption* and *visual fidelity preservation*.

❶ **Watermark Carrier Disruption**. This objective aims at disrupting the watermark carriers identified by the trained watermark carrier encoder, thereby disrupting the watermark signals contained in those carriers. Concretely, we use gradient-based optimization to iteratively update an adversarial image I_{adv} so as to increase the distance between the watermark embedding of I_{adv} and that of the I_w. We write the corresponding loss as:

$$\mathcal{L}_{carrier} = ||\mathcal{E}_{carrier}(\mathcal{T}(I_w)) - \mathcal{E}_{carrier}(\mathcal{T}(I_{adv}))||_2 \quad (1)$$

where $||\cdot||_2$ denotes the L2 distance.

However, while $\mathcal{L}_{carrier}$ pushes the watermark embedding of I_{adv} away from that of the original watermarked image, it may also unintentionally perturb non-carrier image features, which can in turn lead to excessive visual degradation. To address this, we introduce a second loss that encourages I_{adv} to differ from I_w primarily through its watermark embedding, while penalizing I_{adv} that perturb non-carrier features. We leverage the Class Activation Map (CAM) [45] of $\mathcal{E}_{carrier}$ to quantify how extensively a given I_{adv} modifies non-carrier features. CAM highlights, for a given carrier encoder input, the regions that contribute most to the extraction of the watermark embedding. We interpret these regions as the watermark carriers. Concretely, we apply CAM to the watermark carrier encoder input $\mathcal{T}(I_w)$ and obtain a binary indicator map $\mathcal{X}$, where $\mathcal{X}_{i,j} = 1$ denotes that location (i,j) in $\mathcal{T}(I_w)$ is identified as part of the watermark carriers, and $\mathcal{X}_{i,j} = 0$ otherwise. As a result, we want I_{adv} to differ from I_w primarily at locations (i,j) where $\mathcal{X}_{i,j} = 1$. We formalize this with the following loss:

$$\mathcal{L}_{\text{location}} = -\lambda_1 \times \frac{\sum_{i,j} \chi_{i,j} \times |\delta_{i,j}|}{\sum_{i,j} |\delta_{i,j}| + \epsilon} + \lambda_2 \times \frac{\sum_{i,j} (1 - \chi_{i,j}) \times |\delta_{i,j}|}{\sum_{i,j} |\delta_{i,j}| + \epsilon} \quad (2)$$

where λ_1 and λ_2 are scaling weights. ϵ is a small constant to avoid division by zero. $\delta = |\mathcal{T}(I_w) - \mathcal{T}(I_{adv})|$ denotes element-wise magnitude of the modification induced in the watermark carrier encoder's input. $\hat{\delta}_{i,j}$ denotes the modification at location (i,j). The first term in Eq. 2 computes the average modification magnitude on carrier regions (i.e., locations where $\chi_{i,j} = 1$) induced by I_{adv} The second term computes the average modification magnitude on non-carrier regions (i.e., locations where $\chi_{i,j} = 0$) induced by I_{adv}. Minimizing $\mathcal{L}_{\text{location}}$ encourages I_{adv} to perturb the identified carrier regions while penalizing changes to non-carrier features.

❷ **Visual Fidelity Preservation**. This objective aims to enforce visual similarity between I_{adv} and I_w. We employ the Learned Perceptual Image Patch Similarity (**LPIPS**) [46] to quantify perceptual differences between I_w and I_{adv}. LPIPS is a deep learning-based metric that closely models human judgments of image similarity. However, LPIPS may overlook small and sharp changes (e.g., tiny spikes) in I_{adv}. This is because it measures differences using high-level structural features extracted by a neural network, such as textures and edges, rather than fine-grained pixel-level changes. To avoid adding subtle but potentially visible artifacts, we additionally use the L2 norm to directly limit

how much I_{adv} can deviate from I_w at the pixel level. The corresponding losses are defined as follows:

$$\mathcal{L}_{\text{perceptual}} = \alpha_1 \cdot \text{ReLU}\left(\mathcal{L}_{\text{LPIPS}}(I_w, I_{adv}) - \tau_{\text{LPIPS}}\right) + \alpha_2 \cdot \text{ReLU}\left(\|I_w - I_{adv}\|_2 - \tau_{\text{L2}}\right) \quad (3)$$

The ReLU function ensures that a penalty is applied only when the respective loss exceeds its threshold. τ_{LPIPS} and τ_{L2} denote the thresholds for the LPIPS loss and the L2 norm, respectively. α_1 and α_2 are coefficients that balance the contributions of these two loss components.

In summary, given I_w, its non-watermarked counterpart I_{adv} is obtained by solving the following optimization problem:

$$I_{adv} = \arg\min_{I'}(- \mathcal{L}_{\text{carrier}}(I_w, I') + \mathcal{L}_{\text{location}}(I_w, I') + \mathcal{L}_{\text{perceptual}}(I_w, I')) \quad (4)$$

Difference Between Low-perturbation and High-perturbation Schemes. The way to obtain I_{adv} differs between low-perturbation and high-perturbation watermarking schemes (see §2.1). Low-perturbation schemes typically embed watermarks by adding noise-like perturbations to the image. Following this characteristic, *WRATH* generates I_{adv} by selectively modifying individual pixels of I_w. In contrast, high-perturbation schemes introduce structural changes to the image content. Removing these watermarks requires considering inter-pixel links, and modifying pixels independently is insufficient to remove them. To address this, we use convolution-based method proposed in prior work [18]. Leveraging the ability of convolution to capture inter-pixel dependencies and induce structural changes (e.g., blurring the image content), *WRATH* learns a set of content-adaptive linear convolutional kernels which are applied to I_w sequentially to obtain I_{adv}.

Infer Watermark Type via Robustness Profile. Although *WRATH* assumes no knowledge of the underlying watermarking scheme, the type of the scheme (i.e., high-perturbation or low-perturbation) can be inferred from its *robustness profile*. Typically, if a scheme is highly robust against many value-based manipulations (e.g., adding Gaussian noise, JPEG compression, and resizing), it is likely a high-perturbation scheme. Otherwise, it is likely a low-perturbation scheme, as discussed in Appendix A.2.

4.4.2. Watermark Forgery Attack. Given a non-watermarked image, I_{nw}, and a set of images watermarked by the target watermarking scheme, $\{I_w^{(i)}\}_{i=1}^N$, the goal of the forgery attack is to generate an adversarial image, I_{adv}, that is visually similar to I_{nw}, yet is incorrectly recognized by the target watermark detector as watermarked. In this section, we introduce two forgery methods, namely *optimization-based* and *regeneration-based* method.

❶ **Optimization-based Approach.** We directly adapt the optimization method from the removal attack described in §4.3. The key difference between the removal and forgery attacks is that instead of disrupting the carriers identified by the watermark carrier encoder, $\mathcal{E}_{carrier}(\cdot)$, the forgery attack aims to inject watermark signals into these identified carriers. As such, we modify the optimization objective of the removal attack by forcing the watermark embedding, e_w, of I_{nw} to be close to those of $\{I_w^{(i)}\}_{i=1}^N$. Instead of matching I_{adv}'s watermark embedding to that of a single watermarked image, we match it to the watermark embedding averaged across the set $\{I_w^{(i)}\}_{i=1}^N$. This is because some watermarking schemes can be content-adaptive, causing the watermark signal to vary with the image content. Therefore, we use the averaged watermark embedding to capture common watermark signals and minimize the impact of specific image content variations. Formally, the optimization problem for the forgery attack is as follows:

$$I_{adv} = \arg\min_{I'}(\mathcal{L}_{\text{carrier}}(I_{nw}, \{I_w^{(i)}\}_{i=1}^N) + \mathcal{L}_{\text{location}}(I_w, I') + \mathcal{L}_{\text{perceptual}}(I_w, I')) \quad (5)$$

❷ **Regeneration-based Approach.** This method aims to train a model to simulate the behavior of the targeted watermarking scheme to directly add watermark to I_{nw} to generate I_{adv}.[4] Fig. 3 illustrates the overall pipeline of this approach. During the *training phase*, we generate pairs of watermarked and non-watermarked images (the latter obtained using *WRATH* 's removal attack). A trained *Watermark Carrier Encoder*, $\mathcal{E}_{carrier}(\cdot)$, is used to derive the watermark embedding, e_w, from the watermarked image. Concurrently, we train an *Encoder* – built upon a convolutional neural network architecture (CNN) – to extract the image content embedding, e_c. The *Decoder*, implemented as a deconvolutional counterpart to the *Encoder*, is jointly trained to reconstruct the image from the concatenated embeddings e_w and e_c. The training objective is to minimize the L2 loss (i.e., Euclidean distance) between the reconstructed image and the original watermarked image. In the *attack phase*, the trained *Encoder* extracts e_c from a clean, non-watermarked image. An average watermark embedding, $\hat{e}_w$, is computed from a collection of watermarked images using $\mathcal{E}_{carrier}(\cdot)$. Finally, the trained *Decoder* generates a forged image using the concatenated embeddings e_c and $\hat{e}_w$.

4.4.3. Unified Watermark Invalidation Pipeline. We find that optimization-based forgery is more effective for low-perturbation watermarks, whereas regeneration-based method is more effective for high-perturbation ones (§C). This is due to the limitation of linear convolution kernels as they cannot create new structured patterns (e.g., the ring patterns in *Tree-Ring* [10]) if those structures are not already present in the image. In contrast, our CNN-based regeneration method struggles to synthesize the unstructured, noise-like perturbations required for low-perturbation watermarks. We therefore combine both approaches in our full invalidation pipeline (Fig. 3).

4. Since we have black-box assumption on the watermarking scheme, the model structure may be completely different from that used by the watermarking scheme.

5. Evaluation

Our evaluation aims to answer four research questions:

- **RQ1**: How effective is *WRATH* in identifying watermark carriers (§5.2)?
- **RQ2**: How effective is *WRATH* in removing watermarks compared to the state-of-the-art practical attacks tailored for watermark ***removal*** (§5.3)?
- **RQ3**: How effective is *WRATH* in forging watermarks compared to the state-of-the-art practical attack tailored for watermark ***forgery*** (§5.4)?
- **RQ4**: How can the *WRATH* attack be mitigated in practice? (§5.5)?

5.1. Experiment Setup

❶ **Watermarking Schemes.** We evaluate seven state-of-the-art watermarking schemes, including both real-world deployment and academic works. For real-world deployment, we include Amazon Titan's watermarking scheme [4]. We select academic works covering both *semantic*, such as Tree-Ring [10] and Gaussian Shading [11], and *non-semantic* schemes, such as StableSignature [12], DWT-DCT [14], HiDDeN [6], StegaStamp [7]. By analyzing their robustness profile, we categorize Amazon, DWT-DCT, HiDDeN and StableSignature as *low-perturbation* watermarking schemes, while StegaStamp, Tree-Ring, and Gaussian Shading fall under *high-perturbation* schemes (Appendix A.2). This categorization is consistent with findings reported in prior work [29]. We provide categorization details in Tab. 6.

❷ **Image Manipulations.** We construct the robustness profile using a set of 12 image manipulation types, encompassing both *geometry-based* and *value-based* manipulations. Geometry-based manipulations alter the spatial structure of an image, such as rotation and cropping. Value-based manipulations involve changes to pixel intensities, such as brightness adjustment, Gaussian noise addition, and JPEG compression. Following the steps of *Robustness Profile Construction* (§4.2), ***WRATH requires less than 100 queries per evaluated scheme to construct its robustness profile***. We present detailed robustness profiles and the number of queries in Tab. 6.

❸ **Watermarked Dataset.** For each scheme, we generate 1000 watermarked images for training our watermark carrier encoder (§4.3), and 100 unseen images for evaluation. All training images are embedded with the same watermark binary string. We also investigate cases embedding different binary strings across the training set (§5.5.2). Following the findings in prior work [18], the carrier encoder is trained on the images' frequency domain. However, our framework is also applicable to other image domains (§6).

5.2. Effectiveness of Watermark Carrier Encoder

We compare our watermark carrier encoder with three baseline methods that have different capabilities and aim to identify watermark carriers that discriminate between watermarked and non-watermarked images.

Baseline 1: Fragile Manipulation Classifier. We consider a black-box adversary with access only to a set of watermarked images. The adversary first applies fragile manipulations to remove the watermark, treating the resulting image as non-watermarked. Then, the adversary identifies watermark carriers by training a binary classifier to distinguish between watermarked and non-watermarked images. The key difference between this method and *WRATH* is that *WRATH* additionally leverages information from robust manipulations.

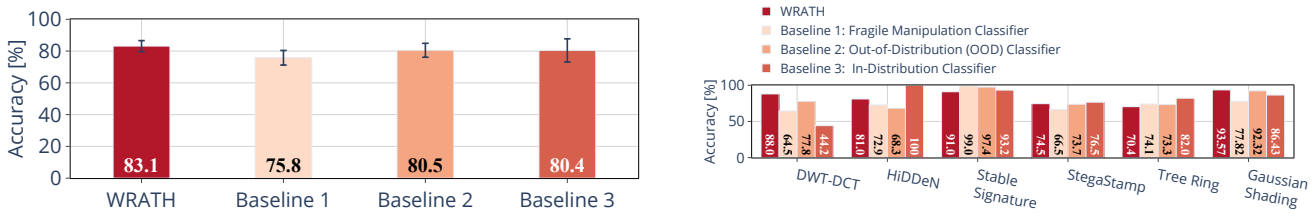
Baseline 2: Out-of-Distribution (OOD) Classifier. We consider a black-box adversary that has access to a set of watermarked images and non-watermarked images from a different source (i.e., *out-of-distribution*), such as real photos or images generated by a different model. The adversary then identifies watermark carriers by training a binary classifier to distinguish between watermarked and OOD non-watermarked images.

Baseline 3: In-Distribution Classifier. We consider a black-box adversary that has access to a set of watermarked images and non-watermarked images from the same source as the watermarked ones (i.e., *in-distribution*). A binary classifier is then trained to distinguish between watermarked and in-distribution non-watermarked images. This method is impractical in real-world because in-distribution non-watermarked images are typically unavailable. However, we employ it as a challenging baseline to compare *WRATH* with an adversary possessing stronger capabilities.

Results. We evaluate the effectiveness of carrier identification using *classification accuracy*, quantifying how reliably the identified carriers distinguish watermarked images from their non-watermarked counterparts. For DWT–DCT, HiDDeN, and StegaStamp, the watermark is added after the image has been generated. The non-watermarked counterparts are the original images prior to embedding. For all other schemes, the watermark is inserted during image generation, which means no non-watermarked originals are available. We therefore use in-distribution non-watermarked images as the counterparts. Due to the lack of in-distribution non-watermarked images for Amazon, we exclude Amazon from this evaluation and focus only on academic schemes.

Fig. 4a illustrates the average accuracy across six schemes for each method evaluated. *WRATH* achieves the highest overall averaged accuracy of 83.1%. Fig. 4b provides a granular breakdown by scheme. Although some baseline methods occasionally outperform *WRATH* in specific scheme, *WRATH* consistently maintains a robust accuracy above 70% across all tested watermarking schemes. This sustained performance validates the strong generalization capabilities of our method compared to the baselines.

We further evaluate the effectiveness of each method in identifying watermark carriers by visualizing each model's Class Activation Maps (CAMs). CAMs highlight the regions within an input image that the model identifies as watermark carriers. The generated CAMs are compared against the ground truth watermark carriers, which are estimated by cal-



(a) Average Classification Accuracy Across Six Schemes (b) Classification Accuracy for Each Scheme

Figure 4. Figure depicts classification accuracy of *WRATH* compared to baselines methods for identifying potential watermark carriers.

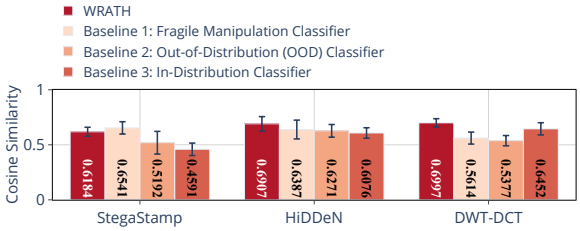


Figure 5. Figure depicts cosine similarity between ground truth watermark carriers and *Class Activation Map* produced by *WRATH* and baselines.

culating the pixel-wise difference between watermarked and original non-watermarked images in the frequency domain. A strong alignment between the CAMs and the ground truth indicates a more accurate localization of the carriers. To quantify this alignment, we utilize cosine similarity between the CAMs and the ground truth. This metric captures both the structural correspondence and the relative magnitude of feature importance. The cosine similarity scores in Fig. 5 cover the schemes for which original non-watermarked images are available for estimating the ground truth carriers. *WRATH* consistently achieves higher similarity scores, thereby demonstrating its ability to localize watermark carriers more accurately. Visualizations of the CAMs can be found in Appendix B.3.

Answer to RQ1: *WRATH* consistently identifies watermark carriers across all evaluated schemes, revealing that many state-of-the-art watermarking designs inadvertently expose their carriers through robustness alone.

5.3. Watermark Removal Attack

We evaluate *WRATH*'s watermark removal capability and compare its performance with *practical* attacks tailored for watermark removal.

Baseline Attacks. As summarized in Tab. 1, existing practical removal attacks primarily targets one of the following watermarking scheme types: i) non-semantic, ii) semantic, or iii) both semantic and non-semantic. In this study, we evaluate representative attacks corresponding to categories i) and iii). There is only one *practical* removal attack targeting semantic schemes [26]. However, this attack is specifically designed for Gaussian Shading [11], and its method cannot be directly applied to other watermarking schemes. Therefore, we exclude category ii) from our evaluation. We list the selected baseline attacks below.

(i) VAEAttack [22] removes the watermark by passing the watermarked image through a variational autoencoder (VAE) to regenerate the image. The regeneration quality is controlled by the "quality" parameter.

(ii) DiffusionAttack [22] removes the watermark by passing the watermarked image through a diffusion model to regenerate the image. The regeneration quality is controlled by the number of diffusion steps.

(iii) UnMarker [18] removes the watermark by first cropping the image by 10% and then perturbing the frequency components of the cropped image. We implement *UnMarker* both with cropping and without cropping as baselines. As *UnMarker* requires careful parameter tuning for each watermarking scheme to achieve its optimal performance, we adopt parameter settings provided by the authors. For schemes not covered by *UnMarker* (i.e., DWT-DCT, and Amazon), we exclude them from the evaluation.

Metrics. We evaluate the watermark removal performance from the perspectives of *success rate* and *perceptual quality*.

(i) Success Rate (SR). For binary-string watermarking schemes, the watermark detector compares the *bit accuracy* against a predefined detection threshold (§2.1). If the bit accuracy falls below the threshold, the image is identified as non-watermarked. We define *SR* as the fraction of watermarked images that are misidentified as non-watermarked after attack. We select the detection threshold for academic watermarking schemes based on the *Equal Error Rate* to balance *false positives* and *false negatives*, where both false positives (i.e., non-watermarked images identified as watermarked) and false negatives (i.e., watermarked images identified as non-watermarked) are considered equally important in the invalidation attack setting. For Amazon, we cannot configure the detection threshold. A detailed discussion of the threshold selection, along with the thresholds selected for each scheme, is provided in Tab. 4. Following prior work [18], an attack is considered ***effective*** if SR exceeds 50%, reflecting that a detector performing no better than random guessing and losses its security advantage.

(ii) Image Quality (IQ). We measure image quality using *LPIPS* [46], which is commonly used as a proxy for perceptual quality. LPIPS falls within the range $[0, 1]$, where lower values correspond to higher perceptual quality. Images are ***perceptually acceptable*** with LPIPS below 0.1 [47].

TABLE 2. EVALUATION OF WATERMARK REMOVAL ATTACKS ACROSS SEVEN SCHEMES. METRICS: BIT ACC↓ / CONF. SCORE↓ / DIST.↑ (SCHEME-SPECIFIC DETECTOR OUTPUTS MEASURED AFTER ATTACK), SR↑ (ATTACK SUCCESS RATE), AND LPIPS↓ (IMAGE PERCEPTUAL QUALITY). FOR AMAZON, WE REPORT CONF. SCORE, MAPPING DETECTOR OUTPUTS { "NOT DETECTED", "LOW", "MEDIUM", "HIGH"} TO {0, 1, 2, 3}. FOR TREE RING, DIST. DENOTES THE DISTANCE BETWEEN THE DECODED AND EMBEDDED PATTERNS. ARROWS DENOTE ATTACKER PREFERENCE: ↑ BETTER, ↓ WORSE. WE BOLD THE BEST ATTACK VALUE IN EACH METRIC COLUMN AND SHADE ENTRIES WHERE SR>50% AND LPIPS<0.1.

Schemes / Removal Attacks	Non-semantic, Low-perturbation												Non-semantic, High-perturbation			Semantic, High-perturbation					
	Amazon			DWT-DCT			HiDDeN			Stable Signature			StegaStamp			Tree Ring			Gaussian Shading		
	Conf. Score ↓	SR↑	LPIPS↓	Bit Acc ↓	SR↑	LPIPS↓	Bit Acc ↓	SR↑	LPIPS↓	Bit Acc ↓	SR↑	LPIPS↓	Bit Acc ↓	SR↑	LPIPS↓	Dist.↑	SR↑	LPIPS↓	Bit Acc ↓	SR↑	LPIPS↓
None	3.00	0%	N.A.	85.78%	7%	N.A.	80.47%	0%	N.A.	97.54%	0%	N.A.	99.43%	0%	N.A.	49.98	0%	N.A.	100%	0%	N.A.
VAE quality=3	**0.00**	**100%**	0.2301	51.77%	**90%**	0.1345	51.53%	84%	0.0360	54.38%	69%	0.3370	99.19%	0%	0.1273	65.82	12%	0.1389	98.78%	0%	0.1295
VAE quality=2	**0.00**	**100%**	0.2474	51.25%	88%	0.1646	47.27%	95%	0.0428	52.17%	83%	0.3729	98.77%	0%	0.1670	69.54	49%	0.1735	97.66%	0%	0.1608
VAE quality=1	**0.00**	**100%**	0.2691	**50.83%**	**90%**	0.1953	**43.63%**	**99%**	0.0549	**51.46%**	**87%**	0.4131	98.04%	0%	0.2065	**72.36**	**81%**	0.2121	95.75%	0%	0.1932
Diffusion steps=50	**0.00**	**100%**	0.1931	51.52%	**90%**	0.1014	54.80%	67%	0.0458	54.25%	68%	0.2820	89.78%	0%	0.1194	69.33	44%	0.1040	99.84%	0%	0.1031
UnMarker (w/ cropping)	-	-	-	-	-	-	56.21%	43%	0.0576	53.37%	73%	0.0400	60.62%	48%	0.0619	70.21	56%	0.0560	**53.60%**	**91%**	0.0161
WRATH (Ours) (w/ cropping)	-	-	-	-	-	-	55.37%	84%	**0.0247**	62.31%	77%	**0.0037**	**58.85%**	83%	0.1388	70.21	70%	0.0987	54.43%	88%	0.0299
UnMarker (w/o cropping)	-	-	-	-	-	-	69.69%	7%	0.0365	56.31%	64%	0.0334	60.39%	46%	**0.0528**	60.66	5%	**0.0476**	99.88%	0%	**0.0046**
WRATH (Ours)* (w/o cropping)	0.06	95%	**0.0185**	55.69%	68%	**0.0220**	56.80%	61%	0.0278	53.81%	75%	0.0086	59.28%	**84%**	0.1717	68.37	51%	0.0901	74.29%	54%	0.2428

Results. Tab. 2 reports bit accuracy, SR, and LPIPS for each attack across seven watermarking schemes. For Amazon, the detector returns categorical confidence ("high", "medium", "low", "not detected",) rather than a numeric bit accuracy. We therefore map these levels to ordinal scores ("high"=3, "medium"=2, "low"=1, "not detected"=0) and report the mean confidence score, where a higher value indicates stronger evidence of watermark presence.

WRATH is the only attack that effectively removes all evaluated semantic and non-semantic watermarks under our SR>50% criterion. Although *VAEAttack* and *DiffusionAttack* have been reported effective on non-semantic schemes, both fail to remove StegaStamp, a high-perturbation non-semantic scheme. While UnMarker has been reported as effective on both semantic and non-semantic schemes, its performance depends heavily on image cropping and it fails on HiDDeN, Tree Ring, and StegaStamp when cropping is disabled. Even with cropping, UnMarker does not *effectively* remove HiDDeN and StegaStamp. For a fair comparison, we evaluate *WRATH* using the same cropping ratio and compare it with UnMarker (w/ cropping). As Tab. 2 shows, cropping further improves *WRATH* 's performance, particularly on HiDDeN, Tree Ring, and Gaussian Shading, with SR increases of 23%, 19%, and 34%, respectively. Although cropping can provide additional gains, *WRATH* 's effectiveness does not depend on cropping. Its success arises from accurately identifying watermark carriers. Notably, ***WRATH removes the watermark while keeping images perceptually acceptable on Amazon, DWT–DCT, HiDDeN, Stable Signature, and Tree Ring.*** Although *WRATH* can introduce visible artifacts when targeting StegaStamp and Gaussian Shading, it remains the only attack that effectively removes these schemes without cropping. Fig. 10 presents sample images attacked by *WRATH*. Overall, the perceptual cost is modest relative to the gain in removal success.

Answer to RQ2: *WRATH* outperforms state-of-the-art *practical* watermark removal attacks, being the ***only*** method that effectively removes ***all*** evaluated schemes, including both semantic and non-semantic, while maintaining high perceptual image quality.

5.4. Watermark Forgery Attack

We evaluate WRATH's watermark forgery capability and compare its performance with practical attacks tailored for watermark forgery.

Baseline Attacks. As summarized in Tab. 1, practical watermark forgery attacks remain limited. Existing practical attacks primarily target either i) non-semantic watermarking schemes or ii) both semantic and non-semantic schemes. To the best of our knowledge, no practical forgery attacks specifically tailored to semantic watermarking schemes have been reported at the time of writing. In this study, we evaluate representative attacks from categories i) and ii) and list the selected baseline attacks below.

(i) Watermark-White-Noise Attack [28] is the only practical forgery attack that generalizes to both semantic and non-semantic schemes. It generates a white-noise image containing the target watermark and adds it to a non-watermarked image. For a fair comparison, we match the perceptual quality of its outputs to those of *WRATH*. We generate watermarked white-noise images by prompting the image generator with "*Please generate a white noise image.*"

(ii) WMCopier Attack [33] trains an unconditional diffusion model to learn the watermark distribution. The estimated watermark is then injected into non-watermarked images through *shallow* inversion, which modifies only the later steps of the diffusion process and therefore largely preserves the original image content. However, this approach does not apply to semantic watermarking schemes, where the watermark must be embedded in the initial noise of the generation process. We use the official implementation

TABLE 3. EVALUATION OF WATERMARK FORGERY ATTACKS ACROSS SEVEN SCHEMES. METRICS: SR↑ (ATTACK SUCCESS RATE), Δ SR↑ (GAIN IN SUCCESS RATE), AND LPIPS↓ (IMAGE PERCEPTUAL QUALITY).

Non-watermarked Image Source		Stable Diffusion V1.4			Stable Diffusion V2			Stable Diffusion V2.1			Stable Diffusion XL			Average ΔSR
Schemes	Forgery Attacks	SR (w/o attack)	SR (ΔSR) (w/ attack)	IQ (LPIPS↓)	SR (w/o attack)	SR (ΔSR) (w/ attack)	IQ (LPIPS↓)	SR (w/o attack)	SR (ΔSR) (w/ attack)	IQ (LPIPS↓)	SR (w/o attack)	SR (ΔSR) (w/ attack)	IQ (LPIPS↓)	
Amazon	WhiteNoiseAtk	5%	2% (-3%)	0.0059	4%	0% (-4%)	0.0051	4%	1% (-3%)	0.0065	1%	2% (+1%)	0.0087	-2.25%
	WRATH	5%	22% (+17%)	0.0063	4%	16% (+12%)	0.0056	4%	11% (+7%)	0.0064	1%	7% (+6%)	0.0082	+10.50%
DWT-DCT	WhiteNoiseAtk	5%	9% (+4%)	0.0467	14%	13% (-1%)	0.0363	9%	14% (+5%)	0.0351	16%	11% (-5%)	0.0323	+0.75%
	WMCopier	5%	18% (+13%)	0.0348	14%	16% (+2%)	0.0296	9%	12% (+3%)	0.0328	16%	20% (+4%)	0.0126	+5.50%
	WRATH	5%	71% (+66%)	0.0426	14%	85% (+71%)	0.0353	9%	93% (+84%)	0.0369	16%	69% (+53%)	0.0303	+68.50%
HiDDeN	WhiteNoiseAtk	0%	1% (+1%)	0.0144	0%	1% (+1%)	0.0122	0%	0% (+0%)	0.0132	0%	0% (+0%)	0.0124	+0.50%
	WRATH	0%	33% (+33%)	0.0125	0%	28% (+28%)	0.0102	0%	33% (+33%)	0.0136	0%	21% (+21%)	0.0121	+28.75%
Stable Signature	WhiteNoiseAtk	25%	17% (-8%)	0.0088	22%	14% (-6%)	0.0377	24%	20% (-4%)	0.0770	15%	12% (-3%)	0.0437	-5.25%
	WMCopier	25%	95% (+70%)	0.2066	22%	89% (+67%)	0.1925	24%	91% (+67%)	0.2827	15%	100% (+85%)	0.1388	+72.25%
	WRATH	25%	77% (+52%)	0.0083	22%	80% (+58%)	0.0380	24%	100% (+76%)	0.0749	15%	80% (+65%)	0.0400	+62.75%
StegaStamp	WhiteNoiseAtk	2%	5% (+3%)	0.0227	4%	2% (-2%)	0.0263	3%	1% (-2%)	0.0101	2%	2% (+0%)	0.0206	-0.25%
	WMCopier	2%	13% (+11%)	0.0594	4%	12% (+8%)	0.0531	3%	21% (+18%)	0.0615	2%	17% (+15%)	0.0279	+13.00%
	WRATH	2%	71% (+69%)	0.0227	4%	69% (+65%)	0.0258	3%	74% (+71%)	0.0096	2%	76% (+74%)	0.0192	+69.75%
Tree Ring	WhiteNoiseAtk	3%	29% (+26%)	0.0819	0%	18% (+18%)	0.1072	0%	22% (+22%)	0.1156	2%	33% (+31%)	0.0982	+24.25%
	WRATH	3%	51% (+48%)	0.0813	0%	41% (+41%)	0.1007	0%	35% (+35%)	0.1118	2%	55% (+53%)	0.0974	+44.25%
Gaussian Shading	WhiteNoiseAtk	1%	89% (+88%)	0.0291	1%	98% (+97%)	0.0228	1%	96% (+95%)	0.0245	1%	100% (+99%)	0.0101	+94.75%
	WRATH	1%	99% (+98%)	0.0260	1%	100% (+99%)	0.0225	1%	100% (+99%)	0.0258	1%	100% (+99%)	0.0102	+98.75%

and pre-trained model weights provided by the WMCopier authors [48], which cover only DWT-DCT, Stable Signature, and StegaStamp. Schemes not covered by WMCopier are therefore excluded from this evaluation.

Non-watermarked Dataset. We generate non-watermarked images using four Stable Diffusion variants: v1.4, v2.0, v2.1, and SD-XL. We then assess how well each attack forges the target watermark on images from these sources.

Metrics. Following the removal-attack evaluation (§5.3), we report *SR* and *IQ*. In the forgery setting, *SR* is the fraction of originally non-watermarked images that are misidentified as watermarked. We also report the gain in success rate, ΔSR, defined as the absolute change in *SR* after the forgery.

Results. Tab. 3 summarizes *SR*, ΔSR, and *IQ* for *WRATH* and baselines across four source generators and seven watermarking schemes. *WRATH* consistently increases *SR*, with average ΔSR of 10.5% (Amazon), 68.5% (DWT-DCT), 28.75% (HiDDeN), 62.75% (Stable Signature), 69.75% (StegaStamp), 44.25% (Tree Ring), and 98.75% (Gaussian Shading). On average, *WRATH* improves ΔSR by 38.7% over White-Noise Attack and by 36.75% over WMCopier. Meanwhile, output images from *WRATH* maintain high perceptual quality, with *IQ* well below 0.1 on average. Fig. 11 presents sample images.

Answer to RQ3: *WRATH* outperforms state-of-the-art forgery attacks, achieving higher success rates across all evaluated schemes while preserving high perceptual quality.

5.5. Practical Defenses Against the *WRATH* Attack

In this section, we analyze potential defenses against the *WRATH* attack. Recall that the attack is enabled by a common design choice in many watermarking schemes: embedding watermark signals into consistent carriers, often with common signal patterns across images. We first validate these observations empirically and then focus on practical defenses that avoid consistent carriers and common watermark signal patterns.

We do not focus on defenses that directly address the *WRATH* vulnerability itself. As shown in Appendix B, our analysis suggests that exploiting *WRATH* without consistent carriers can be more challenging under the approach we evaluated. This observation indicates that mitigating consistent carriers and common watermark signal patterns may already limit the practical exploitation of the vulnerability.

5.5.1. Observation ❶: Consistent Carriers Across Images. We attribute *WRATH*'s effectiveness to a key observation: many watermarking schemes embed watermark signals into largely the same set of image features across different images. If watermark carriers are consistent, *WRATH* can recover them by training on multiple watermarked images.

Measuring Carrier Consistency. We test this hypothesis by estimating watermark carriers for DWT–DCT, HiDDeN, and StegaStamp in the frequency domain and quantifying their similarity. For the other schemes, the original non-watermarked images are unavailable, so their carriers cannot be estimated and are excluded from this analysis. We measure pairwise similarity of carrier frequency bins within each scheme (*intra*-scheme) and across schemes (*inter*-scheme). We measure the similarity using Jaccard index, a higher value indicates that two watermarked images rely on more overlapping frequency bins to carry watermark signals. Our results show **clear carrier consistency**: the average *intra*-scheme Jaccard similarity is 0.52, far exceeding the *inter*-scheme value of 0.16. This difference is statistically significant (Mann–Whitney test, $p < 0.001$), indicating that each scheme repeatedly embeds watermark signals into a consistent set of frequency bins across images.

Defense: Avoid Consistent Carriers. One practical defense is to avoid using consistent carriers across images by making the watermark encoder adapt carrier selection to image content. Instead of embedding watermark signals into fixed

image features, the encoder can select carriers dynamically based on the input image. For example, in neural-network-based watermarking schemes, an attention mechanism can be incorporated into the encoder to condition carrier selection on image content during both the embedding and decoding processes. By varying the carriers across images, such a design reduces the consistency that enables data-driven aggregation of carrier information, thereby making carrier identification more difficult.

To the best of our knowledge, RivaGAN [8] is the only watermarking scheme that leverages an attention mechanism for watermark embedding. Our carrier-consistency analysis suggests that it effectively avoids consistent carriers. Specifically, its *intra*-scheme Jaccard is 0.21, well below the averaged *intra*-scheme Jaccard (0.52) of DWT–DCT, HiDDeN, and StegaStamp. Moreover, its *inter*-scheme Jaccard with HiDDeN and StegaStamp (0.17 and 0.22) is comparable to its *intra*-scheme value, suggesting that RivaGAN does not rely on a fixed set of frequency bins and instead varies carriers across images. Consistent with this observation, *WRATH* appears ineffective against RivaGAN in our evaluation: the attack cannot reliably identify its watermark carriers, and features extracted from RivaGAN-watermarked images are indistinguishable from those of non-watermarked images.

Answer#1 to RQ4: Our results suggest that watermark designers should embed watermarks in diverse, content-dependent carriers to defend against the *WRATH* attack, for example by incorporating an attention mechanism into the watermark encoder and detector.

5.5.2. Observation ❷: Common Watermark Signals Across Images. Beyond using consistent carriers, many schemes embed similar watermark signals, enabling a model to identify carriers by learning common signal patterns.

Measuring Common Watermark Signals. For DWT–DCT, HiDDeN, and StegaStamp, we compute *intra*- and *inter*-scheme cosine similarity over the frequency bins of estimated watermark carriers. Unlike Jaccard, cosine similarity captures both which bins carry watermark signals and how similar the signals are within those bins. Higher values indicate overlapping carriers with aligned patterns. We observe a high *intra*-scheme similarity of 0.91 versus 0.55 *inter*-scheme (Mann–Whitney, $p < 0.001$), showing that residuals from the same scheme ***share nearly identical frequency-domain watermark signal patterns***, while different schemes remain clearly distinct. This forms a scheme-specific "signature" that *WRATH* can learn and replicate for forgery.

Effect of Embedded Bit Strings. Notably, changing the embedded binary string does not significantly alter the watermark signal. Fig. 6a reports *intra*-scheme cosine similarity across images embedded with increasing numbers of distinct binary strings. The similarity remains high and stable as the number of strings grows. This indicates that these schemes embed a common watermark signal largely ***independent of the specific binary string***. To further confirm that common signals enable *WRATH* to learn the signal pattern, we train our carrier encoder on images embedded with between 10 and 50 distinct strings across six academic watermarking schemes. Fig. 6b shows the accuracy of classifying watermarked and non-watermarked images using the learned signal patterns. Increasing the number of distinct strings has little impact on *WRATH* for most schemes, supporting the conclusion that ***common watermark signals allow WRATH to reliably learn carrier patterns***.

Defense: Avoid Common Watermark Signal Patterns. Instead of embedding watermark signals with common patterns across images, watermarking schemes can introduce diversity into watermark signals, for example, by generating key-dependent pseudorandom watermark signals using cryptographic primitives. Such designs prevent common watermark patterns from emerging across images, making it difficult for the carrier encoder to learn shared watermark features and thereby hindering data-driven identification of the underlying watermark pattern.

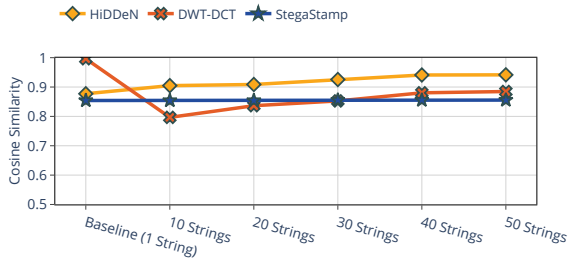
Existing schemes provide examples of this design principle. Gaussian Shading uses secret keys to generate key-dependent watermark signals. As key diversity increases, *WRATH*'s performance degrades: when watermarking 100 images with more than 20 distinct keys, the classification accuracy drops to about 50%. Recent undetectable schemes such as PRC [13] further pursue this direction, aiming to achieve undetectability even when the same key is reused. However, schemes based on cryptographic primitives are highly sensitive to spatial alignment. In our experiments, shifting an image by only a few pixels often suffices to remove the watermark.

Answer#2 to RQ4: Many existing schemes embed common watermark signals across images, enabling *WRATH* to reliably identify and learn carrier patterns. Although current undetectable schemes mitigate this vulnerability by *randomizing* watermark signals, their current designs are extremely fragile to geometric distortions. A key open problem is to build watermarks that remain robust to common image manipulations while thwarting *WRATH*.

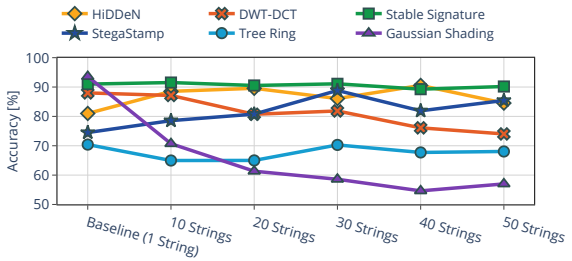
6. Discussion

Domain Generality. *WRATH* is not restricted to the frequency domain. We instantiate it in the frequency domain as prior work [18] shows that robust schemes often embed signals in frequency components, but our framework applies to other domains (e.g., spatial or learned latent spaces). The next step is to systematically explore domains beyond frequency which most effectively reveal watermark carriers.

Consistent Carriers. *WRATH* aims to expose an overlooked vulnerability caused by embedding watermark signals in consistent carriers. Our findings indicate that practitioners frequently adopt this design, intentionally or unintentionally, because they are unaware of the associated risks and thus often do not consider avoiding it. We hope this work encourages the community to avoid using consistent carriers and establish using dynamic carriers as a standard requirement for future watermarks. Additionally, we provide actionable guidelines to help practitioners build effective defenses against the *WRATH* attack.



(a) Cosine similarity of watermark signals across different embedded strings.



(b) Performance of *Watermark Carrier Encoder* trained on images embedded with diverse binary strings.

Figure 6. Impact of varying number of binary strings. (a) Cosine similarity of watermark residuals' frequency bins across images embedded with varying numbers of distinct binary strings. (b) Performance of *WRATH* 's *watermark carrier encoder* trained on images with varying numbers of distinct strings.

API Access Dynamics. Our attack assumes black-box access to a detector API that returns a binary label. Although Amazon's watermark detector exposes a three-level confidence, we adopt the binary setting as a conservative minimal-feedback model, demonstrating that the *WRATH* attack does not rely on detailed detector outputs. However, we envision that richer feedback like confidence levels would enhance attack performance by enabling a more accurate robustness profile. Specifically, incorporating confidence scores would allow us to safely select manipulations that preserve most of the watermark. We could achieve this by choosing intensity levels that yield API outputs with "high confidence". We could also filter out intensity levels that preserve only partial watermark signals yielding API outputs with "low confidence". Improving this robustness profile would help *WRATH* identify more accurate watermark carriers, ultimately making invalidation attacks even more effective (see §4.2).

7. Conclusion

We introduce *WRATH*, a watermark-agnostic, practical, query-based watermark invalidation attack capable of removing and forging watermarks via exploiting watermarking scheme's robustness characteristics. We are pleasantly surprised by the effectiveness of *WRATH* in both removal and forgery attacks. The fact that an essential watermark property – robustness – may be exploited for attacks suggests that other strengths of a watermarking scheme could also be its Achilles heel, and warrants a rethink in watermarking design and security.

8. Ethics Considerations

As misuse of generative AI can cause substantial harm, our intent is to expose risks and guide defenses, not to enable abuse. We follow coordinated disclosure: we have notified Amazon of our findings and provided enough detail to support remediation. We will withhold code and artifacts that could be used to replicate the attack. All experiments rely on publicly available models and datasets, and we avoid generating or disseminating harmful content.

References

[1] "Fake Trump arrest photos: How to spot an AI-generated image," https://www.bbc.com/news/world-us-canada-65069316, 2023.

[2] "Scarlett Johansson on fake AI-generated sex videos: "nothing can stop someone from cutting and pasting my image"," https://www.washingtonpost.com/technology/2018/12/31/scarlett-johansson-fake-ai-generated-sex-videos-nothing-can-stop-someone-cutting-pasting-my-image/, 2018.

[3] "California's New AI Laws Focus on Training Data, Content Transparency," https://www.cooley.com/news/insight/2024/2024-10-16-californias-new-ai-laws-focus-on-training-data-content-transparency, 2024.

[4] "Watermark detection for Amazon Titan Image Generator now available in Amazon Bedrock," 2024, https://aws.amazon.com/about-aws/whats-new/2024/04/watermark-detection-amazon-titan-image-generator-bedrock/.

[5] "Identifying AI-generated images with SynthID," https://deepmind.google/discover/blog/identifying-ai-generated-images-with-synthid/, 2023.

[6] J. Zhu *et al.*, "Hidden: Hiding data with deep networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018.

[7] M. Tancik *et al.*, "Stegastamp: Invisible hyperlinks in physical photographs," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.

[8] K. A. Zhang *et al.*, "Robust invisible video watermarking with attention," *arXiv preprint arXiv:1909.01285*, 2019.

[9] P. Fernandez *et al.*, "Watermarking images in self-supervised latent spaces," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[10] Y. Wen *et al.*, "Tree-rings watermarks: Invisible fingerprints for diffusion images," *Advances in Neural Information Processing Systems*, 2023.

[11] Z. Yang *et al.*, "Gaussian shading: Provable performance-lossless image watermarking for diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[12] P. Fernandez *et al.*, "The stable signature: Rooting watermarks in latent diffusion models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[13] S. Gunn *et al.*, "An undetectable watermark for generative image models," *arXiv preprint arXiv:2410.07369*, 2024.

[14] A. Al-Haj, "Combined dwt-dct digital image watermarking," *Journal of computer science*, 2007.

[15] P. Fernandez *et al.*, "Watermarking images in self-supervised latent spaces," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[16] H. Ci *et al.*, "Ringid: Rethinking tree-ring watermarking for enhanced multi-key identification," 2024.

[17] K. Arabi *et al.*, "Hidden in the noise: Two-stage robust watermarking for images," 2025. [Online]. Available: https://arxiv.org/abs/2412.04653

[18] A. Kassis *et al.*, "Unmarker: A universal attack on defensive watermarking," *arXiv preprint arXiv:2405.08363*, 2024.

[19] Y. Hu *et al.*, "A transfer attack to image watermarks," *arXiv preprint arXiv:2403.15365*, 2024.

[20] X. Li, "Diffwa: Diffusion models for watermark attack," in *2023 International Conference on Integrated Intelligence and Communication Systems (ICIICS)*, 2023.

[21] Y. Hu, Z. Jiang, M. Guo, and N. Gong, "Stable signature is unstable: Removing image watermark from diffusion models," *arXiv preprint arXiv:2405.07145*, 2024.

[22] X. Zhao *et al.*, "Invisible image watermarks are provably removable using generative ai," *Advances in Neural Information Processing Systems*, 2024.

[23] Z. Jiang *et al.*, "Evading watermark based detection of ai-generated content," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023.

[24] H. Liang, T. Li, and J. Sun, "A baseline method for removing invisible image watermarks using deep image prior," *arXiv preprint arXiv:2502.13998*, 2025.

[25] J. Lin and M. Juarez, "A crack in the bark: Leveraging public knowledge to remove tree-ring watermarks," *arXiv preprint arXiv:2506.10502*, 2025.

[26] D. Z. Lee, H. Fang, H. Wang, and E.-C. Chang, "Removal attack and defense on ai-generated content latent-based watermarking," *arXiv preprint arXiv:2509.11745*, 2025.

[27] N. Lukas *et al.*, "Leveraging optimization for adaptive attacks on image watermarks," *arXiv preprint arXiv:2309.16952*, 2023.

[28] M. Saberi *et al.*, "Robustness of ai-image detectors: Fundamental limits and practical attacks," *arXiv preprint arXiv:2310.00076*, 2023.

[29] Y. Liu *et al.*, "Image watermarks are removable using controllable regeneration from clean noise," *arXiv preprint arXiv:2410.05470*, 2024.

[30] B. An *et al.*, "Waves: Benchmarking the robustness of image watermarks," *arXiv preprint arXiv:2401.08573*, 2024.

[31] R. Wang *et al.*, "Watermark faker: towards forgery of digital image watermarking," in *2021 IEEE International Conference on Multimedia and Expo (ICME)*, 2021.

[32] M. Kutter *et al.*, "Watermark copy attack," in *Security and Watermarking of Multimedia Contents II*, 2000.

[33] Z. Dong, C. Shuai, Z. Ba, P. Cheng, Z. Qin, Q. Wang, and K. Ren, "Wmcopier: Forging invisible image watermarks on arbitrary images," *arXiv preprint arXiv:2503.22330*, 2025.

[34] ——, "Imperceptible but forgeable: Practical invisible watermark forgery via diffusion models," *arXiv e-prints*, 2025.

[35] C. Zhu, Z. Li, R. Yang, R. Birke, P.-Y. Chen, T.-Y. Ho, and L. Y. Chen, "Optimization-free universal watermark forgery with regenerative diffusion models," *arXiv preprint arXiv:2506.06018*, 2025.

[36] V. Kinakh, B. Pulfer, Y. Belousov, P. Fernandez, T. Furon, and S. Voloshynovskiy, "Evaluation of security of ml-based watermarking: Copy and removal attacks," in *2024 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2024.

[37] G. Li, Y. Chen, J. Zhang, J. Li, S. Guo, and T. Zhang, "Warfare: Breaking the watermark protection of ai-generated content," *arXiv preprint arXiv:2310.07726*, 2023.

[38] Z. Ba, Y. Zhang, P. Cheng, B. Gong, X. Zhang, Q. Wang, and K. Ren, "Robust watermarks leak: Channel-aware feature extraction enables adversarial watermark manipulation," *arXiv preprint arXiv:2502.06418*, 2025.

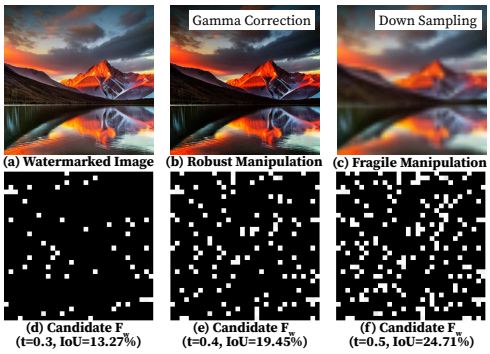


Figure 7. Figure illustrates how adjusting the threshold value t, which defines what constitutes a "significant" change under robust and fragile manipulations, can substantially affect the resulting candidate carrier set. (a) A watermarked image. (b) Image after applying a robust manipulation (e.g., gamma correction). (c) Image after applying a fragile manipulation (e.g., downsampling). (d)–(f) Candidate sets computed under different threshold values t (see Appendix B.1 for details), along with their corresponding Intersection over Union (IoU) percentages. As shown, varying t leads to significantly different candidate carrier set.

[39] G. Li, Y. Chen, J. Zhang, J. Li, S. Guo, and T. Zhang, "Towards the vulnerability of watermarking artificial intelligence generated content," 2023.

[40] A. Müller *et al.*, "Black-box forgery attacks on semantic watermarks for diffusion models," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025.

[41] P. Yang, H. Ci, Y. Song, and M. Z. Shou, "Can simple averaging defeat modern watermarks?" *Advances in Neural Information Processing Systems*, 2024.

[42] A. Jain, Y. Kobayashi, N. Murata, Y. Takida, T. Shibuya, Y. Mitsufuji, N. Cohen, N. Memon, and J. Togelius, "Forging and removing latent-noise diffusion watermarks using a single image," *arXiv preprint arXiv:2504.20111*, 2025.

[43] F. Schroff *et al.*, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[44] Y. Bengio *et al.*, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009.

[45] J. Gildenblat *et al.*, "Pytorch library for cam methods," https://github.com/jacobgil/pytorch-grad-cam, 2021.

[46] R. Zhang *et al.*, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[47] S. H. Park *et al.*, "Perception-oriented single image super-resolution using optimal objective estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023.

[48] Z. Dong *et al.*, "Wmcopier github," https://github.com/holdrain/WMCopier.

[49] S. Lu *et al.*, "Robust watermarking using generative priors against image editing: From benchmarking to advances," *arXiv preprint arXiv:2410.18775*, 2024.

Appendix A. Supplementary: Watermarking Schemes

Tab. 6 summarizes each watermarking scheme's type (e.g., *low-perturbation* or *high-perturbation*, *semantic* or *non-semantic*), detection threshold, robustness profile, and

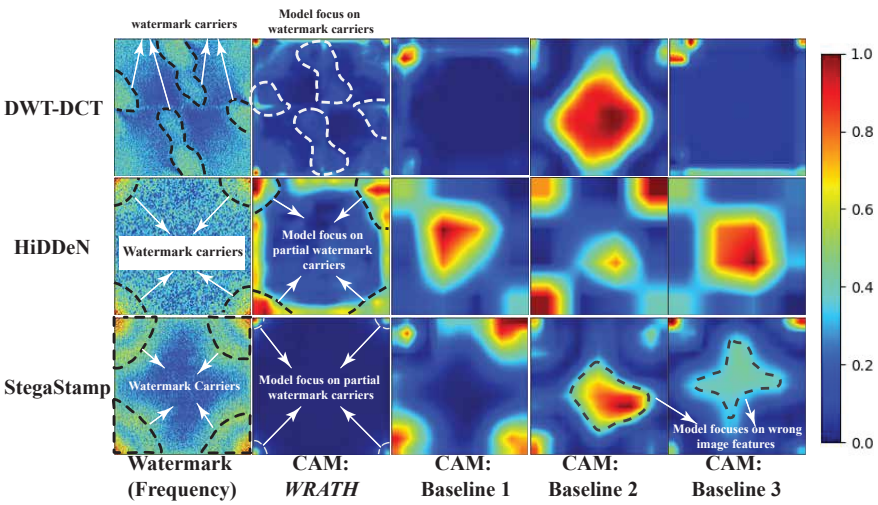


Figure 8. Figure depicts the Class Activation Maps (CAMs) produced by *WRATH* and baselines (§5.2) on DWT-DCT, HiDDeN, and StegaStamp. Warm-colored regions indicate where the model considers as the potential watermark carrier. Notably, the CAMs produced by *WRATH* most closely align with the ground truth watermark carriers.

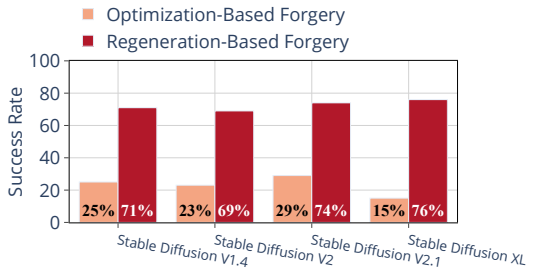


Figure 9. Figure depicts the forgery success rate (*SR*) against StegaStamp for *WRATH*'s perturbation-based and regeneration-based attacks.

TABLE 4. TABLE REPORT THE SELECTED WATERMARK DETECTION THRESHOLDS AND THE CORRESPONDING FALSE POSITIVE RATES (FPR) AND FALSE NEGATIVE RATES (FNR) FOR TWO THRESHOLD SELECTION METHODS: EQUAL ERROR RATE (EER) AND FIXED FAR AT 0.01.

Schemes	Select@ EER			Select@ FPR=0.01		
	Thresh.	FNR	FPR	Thresh.	FNR	FPR
DWT-DCT	56.92%	0.07	0.06	61.54%	0.08	0.01
StableSignature	58.30%	0.11	0.14	68.75%	0.19	0.01
HiDDeN	56.67%	0.00	0.00	54.00%	0.00	0.02
StegaStamp	59.38%	0.00	0.00	57.00%	0.00	0.01
Tree-Ring	69.69	0.00	0.00	69.77	0.00	0.01

TABLE 5. TABLE SUMMARIZES THE CORRELATION BETWEEN WATERMARK CARRIER REGION SIZE AND TRAINING LOSS ACROSS FIVE WATERMARKING SCHEMES.

Watermarking Scheme	Correlation Coef.
DWT-DCT	-0.97
HiDDeN	-0.85
Stable Signature	-0.95
StegaStamp	-0.93
Tree Ring	-0.99

the number of queries *WRATH* requires to obtain it. For each scheme, *WRATH* requires fewer than 100 detector queries to construct the profile, demonstrating its practicality.

A.1. Detection Threshold Selection

We select detection thresholds for academic watermarking schemes using the Equal Error Rate (EER), which balances false positives and false negatives. In the invalidation-attack setting, we treat false positives and false negatives as equally important because they enable different adversarial goals. A false negative (a watermarked image classified as non-watermarked) corresponds to a successful removal attack, whereas a false positive (a non-watermarked image classified as watermarked) corresponds to a successful forgery attack. In contrast, conventional practice selects the threshold to achieve a fixed false positive rate (FPR) at 0.01 [10], [12], [13], [18], [29]. This strategy is motivated by deployment concerns: it prioritizes limiting false accusations by ensuring the detector does not label too many non-watermarked images as watermarked.

Tab. 4 reports the selected thresholds, along with the resulting FPR and false negative rate (FNR) under both threshold selection strategies. While widely used, the fixed-FPR approach can yield overly conservative thresholds and thus inflate the false negative rate (FNR) when the detector does not clearly separate watermarked from non-watermarked images. In such cases, tightening the threshold to keep false positives below 1% inevitably causes more true watermarked images to fall below the decision boundary, raising FNR. For example, with Stable Signature, enforcing FPR = 0.01 leads to FNR = 0.19. This high FNR is problematic because many watermarked images are misclassified as non-watermarked, making removal attacks easier to succeed. Moreover, when the detector is capable of clearly distinguishing between watermarked and non-watermarked images, the EER-based

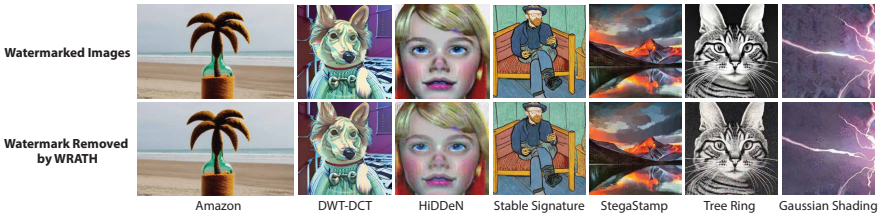


Figure 10. The figure shows examples of watermarked images and their corresponding outputs after applying the *WRATH* removal attack.

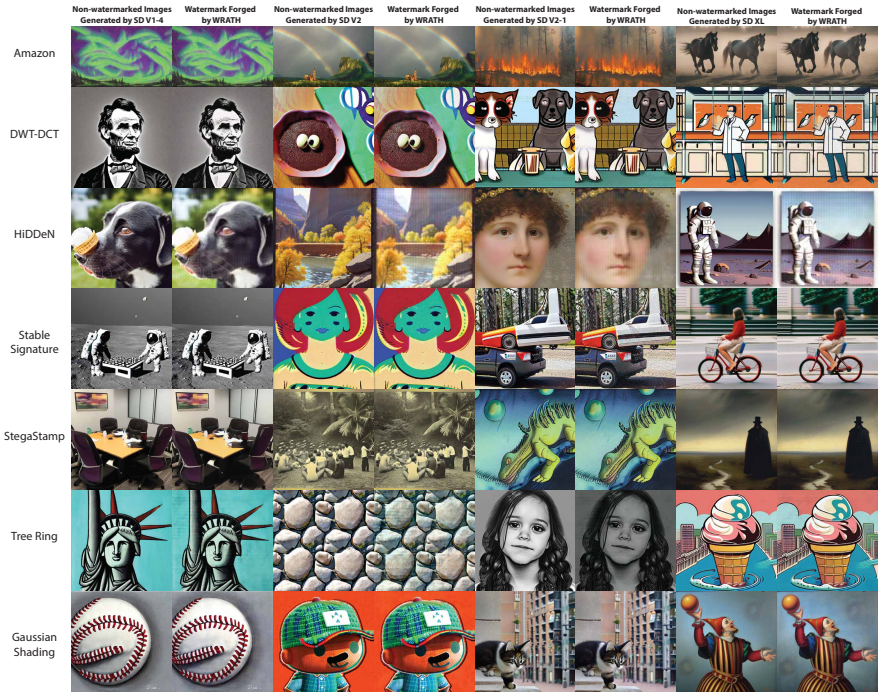


Figure 11. The figure shows examples of non-watermarked images and their corresponding outputs after applying the *WRATH* forgery attack.

threshold often results in negligible FPR and FNR, outperforming the fixed-FAR approach, which by design maintains a nonzero FPR (e.g., 0.01). This is evident in schemes such as HiDDeN, StegaStamp, and Tree-Ring.

Takeaway*: threshold selection should not reflexively follow the fixed-FPR convention. It should be chosen to match the operating goals of the application (e.g., balancing errors via EER versus limiting FPR for deployment).*

A.2. Inferring Watermark Types via Robustness

We observe that the watermark type can be inferred from a scheme's robustness profile. Specifically, if a scheme remains highly robust against many value-change-based manipulations (e.g., Gaussian noise addition, JPEG compression, resize), it is likely a high-perturbation scheme. For example, StegaStamp, Tree-Ring and Gaussian Shading are primarily removable through geometric transformations such as rotation, cropping, and shearing. This is because high-perturbation schemes tend to embed watermarks by manipulating the image's low-frequency components, and removing such watermarks typically requires structural changes. As a result, simple pixel value modifications or scaling operations are generally ineffective [49].

Appendix B. Supplementary: Watermark Carrier Encoder

We present supplementary materials that justify the design of the *watermark carrier encoder* (§4.3).

B.1. Identifying Watermark Carriers Using Non-Learning-Based Method.

We investigate the feasibility of identifying image features that are perturbed by a fragile manipulation but remain unaffected by a robust manipulation, *without relying on data-driven methods*. Let I_w denote a watermarked image, I_{robust} the robustly manipulated version, and $I_{fragile}$ the version subjected to fragile manipulation. In this experiment, we transform each image into its frequency domain via FFT, yielding feature representations F_w, F_{robust} and $F_{fragile}$.

TABLE 6. TABLE SUMMARIZES EACH WATERMARKING SCHEME'S TYPE (E.G., LOW-PERTURBATION OR HIGH-PERTURBATION), DETECTION THRESHOLD, ROBUSTNESS PROFILE, AND THE NUMBER OF QUERIES *WRATH* REQUIRES TO OBTAIN THE CORRESPONDING PROFILE.

Scheme	Type	Detection Threshold	Robustness Profile				# of queries
			Robust Manip.		Fragile Manip.		
Amazon	Low-perturbation;	-	Center crop: 0.9 Horizontal shear: 0.1,-0.1 Vertical shear: 0.1,-0.1 Gaussian noise: 10 Brightness: 1.2, 0.5	Contrast: 3.6, 0.4 Gamma correction: 0.4, 1.4 JPEG: 90 Resize: 0.9	Rotate: 25 Center crop: 0.6 Horizontal shear: 0.9,-1.1 Vertical shear: 0.4,-0.4 Gaussian noise: 20 Brightness: 4.8, 0.4	Contrast: 0.3 Gamma correction: 0.3, 1.5 JPEG: 20 Resize: 0.8 Median filter: 3 Gaussian blur: 1	84
DWT-DCT	Low-perturbation; Non-semantic scheme	56.92%	Vertical shear: 0.1,-0.1 Gaussian noise: 10 Brightness: 1.2, 0.9 Contrast: 1.2, 0.9 Gamma correction: 0.9, 1.5 Resize: 0.9		Rotate: 3 Center crop: 0.9 Horizontal shear: 0.1,-0.1 Vertical shear: 1,-1.8 Gaussian noise: 21 Brightness: 3.3, 0.6	Contrast: 3.2, 0.6 Gamma correction: 0.3 JPEG: 80 Resize: 0.5 Median filter: 3 Gaussian blur: 1	85
StableSignature	Low-perturbation; Non-semantic scheme	58.30%	Rotate: 1 Center crop: 0.9 Horizontal shear: 0.1,-0.1 Vertical shear: 0.1,-0.1 Gaussian noise: 10 Brightness: 3.6, 0.9	Contrast: 3.6, 0.9 Gamma correction: 0.5, 1.5 JPEG: 50 Resize: 0.9 Median filter: 3 Gaussian blur: 1	Rotate: 34 Center crop: 0.4 Horizontal shear: 1.2,-1.2 Vertical shear: 1.3,-0.8 Gaussian noise: 56 Brightness: 0.2	Contrast: 0.1 Resize: 0.2 Median filter: 13 Gaussian blur: 2.6	83
HiDDeN	Low-perturbation; Non-semantic scheme	56.67%	Rotate: 1 Center crop: 0.9 Horizontal shear: 0.1,-0.1 Vertical shear: 0.1,-0.1 Brightness: 0.5, 1.2	Contrast: 0.5, 3.6 Gamma correction: 0.5, 1.5 Resize: 0.9 Median filter: 3 Gaussian blur: 1	Rotate: 17 Center crop: 0.5 Horizontal shear: 0.6,-0.7 Vertical shear: 1.1,-1.2 Gaussian noise: 17 Brightness: 5	JPEG: 90 Resize: 0.3 Median filter: 7 Gaussian blur: 1.6	92
StegaStamp	High-perturbation; Semantic scheme	59.38%	Gaussian noise: 155 Brightness: 0.5, 3.6 Gaussian blur: 3.5 Median filter: 13	Contrast: 0.5, 3.6 Gamma correction: 0.5, 1.5 JPEG: 50 Resize: 0.5	Rotate: 10 Center crop: 0.8 Horizontal shear: 0.2,-0.2 Vertical shear: 0.2,-0.2		59
Tree-Ring	High-perturbation; Semantic scheme	69.69	Brightness: 0.5, 3.6 Gaussian blur: 3 Median filter: 13	Contrast: 0.5, 3.6 Gamma correction: 0.5, 1.5 JPEG: 50 Resize: 0.5	Rotate: 23 Center crop: 0.9 Horizontal shear: 1,-1.5	Vertical shear: 1,-1 Gaussian noise: 82	61
Gaussian Shading	High-perturbation; Semantic scheme	57.80%	Gaussian noise: 155 Brightness: 0.5, 3.6 Median filter: 13 Gaussian blur: 3.6	Contrast: 0.1, 3.6 Gamma correction: 0.5, 1.5 JPEG: 50 Resize: 0.5	Rotate: 5 Center crop: 0.9 Horizontal shear: 0.2,-0.2 Vertical shear: 0.3,-0.2		53

We compute the feature difference and apply a threshold t to isolate meaningful or significant changes. Robust-invariant features are defined as

$$\Delta F_{robust} = \{(F_{robust} - F_w) | (F_{robust} - F_w) < t\}$$

while fragile-sensitive features are given by

$$\Delta F_{fragile} = \{(F_{fragile} - F_w) | (F_{fragile} - F_w) > t\}$$

The intersection $\Delta F_{rosbut} \cap \Delta F_{fragile}$ defines the candidate watermark carrier set. To quantify its size, we compute the Intersection over Union (IoU) as

$$\frac{\Delta F_{rosbut} \cap \Delta F_{fragile}}{\Delta F_{rosbut} \cup \Delta F_{fragile}} \times 100\%$$

However, this region is highly sensitive to the choice of threshold t, as depicted in Fig. 7. Due to the absence of ground-truth watermark carriers, selecting an optimal threshold is inherently difficult. Consequently, ***instead of relying solely on this heuristic method, we adopt data driven method to locate the watermark carriers***.

B.2. Correlation Between Training Loss and Carrier Set Size

We observe a strong correlation between the training *triplet loss* and the size of the candidate carrier set, as estimated by the IoU percentage using the aforementioned method. Specifically, smaller IoU values tend to correspond to higher training losses. Although IoU depends on a threshold and is unreliable for measuring the absolute size of candidate sets, we use it here only as a relative proxy across triplets. The same threshold is applied uniformly to all triplets, so our conclusion relies on the ordering rather than the absolute values. We compute the correlation coefficient between the IoU and *Triplet Loss* across five different watermarking schemes. The results, summarized in Tab. 5, highlight consistently negative correlation coefficients with magnitudes ranging from 0.85 to 0.99, indicating a strong inverse relationship. ***This motivates the adoption of a Curriculum Learning strategy to support model training.***

B.3. Class Activation Map of Carrier Encoder

Fig. 8 depicts the CAMs produced by our watermark carrier encoder and baseline models on DWT-DCT, HiDDeN, and StegaStamp (§5.2). Visually, our method identifies the carriers more precisely than the baseline methods, particularly in the case of DWT-DCT. In contrast, the baselines struggle to identify the correct watermark carriers.

Appendix C. Supplementary: Invalidation Attack

We compare the success rates of optimization- and regeneration-based forgery attacks against the high-perturbation scheme StegaStamp in Fig. 9. The regeneration-based attack achieves an average *SR* of 72.5% across four non-watermarked image sources, significantly outperforming the perturbation-based attack, which achieves only 23%. These results highlight the effectiveness of regeneration-based forgery against high-perturbation schemes.

Appendix D. Meta-Review

The following meta-review was prepared by the program committee for the 2026 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

D.1. Summary

The authors provide the first approach to blend watermark removal and forgery within a single pipeline. When tested against a simulacra of the Amazon watermarking API, the authors were able to demonstrate promising results. This work shines an interesting light on the tension between robustness and security, driven by the potential for robustness to drive information leakages.

D.2. Reasons for Acceptance

1) This work has the potential to enhance the communities understanding of watermarking dynamics, and provides a watermark for future offensive and defensive works within this space.
2) All reviewers agreed that the work was well motivated, and comprehensively explored an area with important implications for digital security.
3) Tested experiments demonstrated the utility of the approach in a setting with strong parallels to real world environments.

D.3. Noteworthy Concerns

1) A reliance upon consistency in the watermark carrier location, which may not be a practical assumption, or may lead to easy opportunities for defeating this technique.
2) Reviewers noted potential divergence between experimental settings and potential deployment scenarios. These include, but are not limited to, the implications of any API access dynamics, and broader concerns about potential changes if additional API options become available, as the design choices are heavily predicated upon Amazon's API.
3) The claim to novelty is, in part, due to unifying removal and forgery as a single process, however, the arguments contained within the paper were not necessarily convincing about why there was inherent utility in a merged pipeline.

Appendix E. Response to the Meta-Review

1) *WRATH* aims to expose an overlooked vulnerability caused by embedding watermark signals in consistent carriers. Our findings indicate that practitioners frequently adopt this design, intentionally or unintentionally, because they are unaware of the associated risks and thus often do not consider avoiding it. We hope this work encourages the community to avoid using consistent carriers and establish using dynamic carriers as a standard requirement for future watermarks. Additionally, we provide actionable guidelines to help practitioners build effective defenses against the *WRATH* attack (see §5.5).
2) We intentionally evaluate *WRATH* in a minimal-feedback setting where the API provides only binary output. The *WRATH* attack remains highly effective with minimal information and this highlights the severity of the underlying vulnerability. Furthermore, we envision that richer feedback like confidence levels would enhance attack performance by enabling a more accurate robustness profile. Specifically, incorporating confidence scores would allow us to safely select manipulations that preserve most of the watermark. We could achieve this by choosing intensity levels that yield API outputs with "high confidence". We could also filter out intensity levels that preserve only partial watermark signals yielding API outputs with "low confidence". Improving this robustness profile would help *WRATH* identify more accurate watermark carriers, ultimately making invalidation attacks even more effective (see §4.2).
3) We acknowledge that utility gains of unifying removal and forgery as a single process could be limited. However, the core novelty of *WRATH* lies in exposing a previously overlooked vulnerability in recent watermarking systems, where robustness itself can become a source of information leakage. This vulnerability is amplified by a common design choice of using consistent watermark carriers. Our unified attack naturally emerges from these findings. It successfully compromises security objectives that prior work treated separately under varying attack assumptions and across fundamentally distinct domains.