WATERFLOW: LEARNING FAST & ROBUST WATERMARKS USING STABLE DIFFUSION

Vinay Shukla
UCLA
vshukla@g.ucla.edu

Prachee Sharma
Netflix
prachees@alumni.cmu.edu

Ryan Rossi
Adobe Research
ryrossi@adobe.com

Sungchul Kim
Adobe Research
sukim@adobe.com

Tong Yu
Adobe Research
tyu@adobe.com

Aditya Grover
UCLA
adgrover@g.ucla.edu

ABSTRACT

The ability to embed watermarks in images is a fundamental problem of interest for computer vision, and is exacerbated by the rapid rise of generated imagery in recent times. Current state-of-the-art techniques suffer from computational and statistical challenges such as the slow execution speed for practical deployments. In addition, other works trade off fast watermarking speeds but suffer greatly in their robustness or perceptual quality. In this work, we propose WaterFlow (WF), a fast and extremely robust approach for high fidelity visual watermarking based on a learned latent-dependent watermark. Our approach utilizes a pretrained latent diffusion model to encode an arbitrary image into a latent space and produces a learned watermark that is then planted into the Fourier Domain of the latent. The transformation is specified via invertible flow layers that enhance the expressivity of the latent space of the pre-trained model to better preserve image quality while permitting robust and tractable detection. Most notably, WaterFlow demonstrates *state-of-the-art performance on general robustness* and is the *first method capable of effectively defending against difficult combination attacks*. We validate our findings on three widely used real and generated datasets: MS-COCO, DiffusionDB, and WikiArt.

1 INTRODUCTION

Watermarking techniques for digital content have been extensively studied over the past decades, focusing on embedding minimal modifications within images to assert their origins and authenticity Cox et al. (2002). Given recent advances in the ease, fidelity, and speed of image generation technology, another growing use case of visual watermarking is to attribute ownership to synthetic imagery and prevent misuse of generative models for deceptive and unauthorized purposes such as creating misleading or fake content Franceschelli & Musolesi (2022). Hence, our goal is to design a deployable general-purpose solution for visual watermarking in real-time that is broadly applicable to real and synthetic images.

The key challenge for visual watermarking is to preserve the quality of the original image and ensure that the watermark can be reliably detected even in the presence of image distortions and transformations, such as lossy compression, rotations, and adversarial attacks. Early approaches to visual watermarking focused on embedding a visually imperceptible signature through embedding techniques that operate in the frequency domain Kundur & Hatzinakos (1998), texture-rich regions Bender et al. (1996), or within the least significant bits Wolfgang & Delp (1996).

In recent years, numerous deep learning approaches have been developed for visual watermarking, each differing in architectural design, data processing techniques, and learning objectives Zhu et al. (2018); Luo et al. (2020); Zhang et al. (2019b). However, these methods are not well-equipped to counter emerging generative-based attacks Zhao et al. (2023a); Ballé et al. (2018); Cheng et al. (2020), which leverage Variational Autoencoders (VAEs) or diffusion models to regenerate images and remove embedded watermarks.

A related line of research explores watermarking model outputs by embedding watermarks in training datasets Zhao et al. (2023b) or modifying the sampling process Fernandez et al. (2023). More recently, diffusion models have been investigated as a direct watermarking mechanism. Tree-Ring Wen et al. (2024) introduces a method that embeds circular watermarks in the frequency domain of image latents, ensuring imperceptibility in image space. Stable Signature Fernandez et al. (2023) further suggests that diffusion models can serve as effective watermarking tools for defending against generative-based attacks.

ZoDiac Zhang et al. (2024) optimizes a latent-space vector within Stable Diffusion allowing for. This method demonstrated significantly greater robustness against generative attacks and is applicable to real-world images. Despite its general applicability, ZoDiac has a slow watermarking speed, requiring optimization for each individual latent image. It also struggles to defend against aggressive attacks that apply multiple perturbations simultaneously.

In this work, we introduce WaterFlow (WF), a novel approach to high-fidelity visual watermarking that is fast, robust, and adaptable to both real and synthetic image domains. WaterFlow operates in three key phases: training, watermark embedding, and detection. During the initial training phase, we learn a lightweight flow model that generates a latent-dependent watermark. This training process is performed only once. For watermarking, we first derive a watermark from the latent vector of an image produced by a pretrained generative model, such as Stable Diffusion in our experiments. The latent vectors are then transformed into the frequency domain and processed through our learned mapping to produce a latent-dependent mark. Finally, this mark is embedded into the original latent representation, generating a new image.

Our main contributions are as follows:

- We propose a visual watermarking method called WaterFlow, which preserves image fidelity while having fast watermarking speed and *achieving state-of-the-art general robustness against adversarial attacks*.
- We develop a learned mapping that dynamically balances watermark detectability and image quality, enabling a more flexible and adaptive watermarking approach.
- We introduce the *first watermarking method in the literature that withstands complex combination attacks* and provides state-of-the-art defense against generative-based attacks.

2 Related Work

Image Watermarking: Watermarking images is crucial to prevent misuse of generative models for deceptive and unauthorized purposes such as creating misleading or fake content Hu et al. (2024). Traditional approaches have largely utilized frequency decomposition techniques Bors & Pitas (1996); Xia et al. (1998); Urvoy et al. (2014). These methods are favored due to their robustness to standard image manipulations, including translations, rotations, and resizing, thus ensuring the durability of watermarks against such transformations.

In response to the emergence of deep neural networks (DNNs), novel learning-based watermarking techniques have been developed Hayes & Danezis (2017); Zhu et al. (2018); Tancik et al. (2019) that jointly train end-to-end watermarking models to maximize transmission and robustness. To further enhance the performance and robustness, Generative Adversarial Network (GAN)-based approaches have been introduced Zhang et al. (2019b;a); Huang et al. (2023); Ma et al. (2022). Other works extend this paradigm by leveraging invertible neural networks to simultaneously perform encoding and decoding Ma et al. (2022). While effective against conventional attacks, this class of method struggle to protect images from modern generative attacks, which employ autoencoders and diffusion-based models to "redraw" images Zhao et al. (2023a).

Watermarking for Diffusion Model: Recent work on diffusion-based watermarking has focused on enabling these models to generate watermarked images directly by fine-tuning them with datasets containing watermarked images Wang et al. (2023); Cui et al. (2023); Zhao et al. (2023b).

On the other hand, Fernandez et al. (2023) proposed Stable Signature, which consists of three steps: (1) pretraining a watermark encoder-decoder on images, (2) fine-tuning the LDM while freezing the encoder and updating only the decoder to enforce a fixed signature, and (3) generating images with

embedded signatures via the LDM's decoder. Min et al. (2024) introduced WaDiff, an extension of Stable Signature that trains an image encoder-decoder for message retrieval and applies a consistency loss between watermarked and non-watermarked samples. However, both methods are limited to generated images and do not support watermarking real-world images.

Recent advances use diffusion models for watermarking all images, embedding watermarks in their latent space Wen et al. (2024); Yang et al. (2024); Tan et al. (2024); Lei et al. (2024). Wen et al. (2024) introduces Tree Ring watermarks, embedding marks in the initial latent and detecting them via DDIM inversion. These methods resist diffusion-based attacks Zhao et al. (2023a) but struggle with complex combination attacks, stable diffusion-based threats, and geometric transformations. Zhang et al. (2024) proposes ZoDiac, which builds of Tree Rings and optimizes a latent for perceptual quality. This method suffers greatly from slow watermarking speed and weak defense against complex attacks.

Diffusion Models and DDIM: We introduce the basic background of diffusion models and, more specifically, DDIM sampling Ho et al. (2020); Song et al. (2020b); Dhariwal & Nichol (2021). A forward diffusion process is made up of T, where a Gaussian noise vector is gradually mapped to a real data point $x_0 \sim q(x_0)$, and $q(x_t)$ represents the real data distribution:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{1}$$

As shown, the transformation is guided by the Markovian assumption, where $\beta_t \in (0, 1)$ is the variance at step t. The closed-form solution for this sampling is:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \tag{2}$$

and $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$.

For the reverse diffusion process, DDIM Song et al. (2020a) sampling is an efficient deterministic strategy (as opposed to DDPM Ho et al. (2020)). Starting from a Gaussian vector $x_T \sim \mathcal{N}(0, 1)$ to an image $x_0 \sim q(x)$. For each de-noise step, a learned noise predictor ϵ_θ estimates the noise $\epsilon_\theta(x_t)$ added to x_0. We can derive the estimation of x_0 as:

$$\tilde{x}_0^t = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t)}{\sqrt{\bar{\alpha}_t}}. \tag{3}$$

Then, we add the estimated noise to $\tilde{x}_0^t$ to find x_{t-1}:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\tilde{x}_0^t + \sqrt{1-\bar{\alpha}_{t-1}}\epsilon_\theta(x_t). \tag{4}$$

This recursive denoising process transitions from x_T to x_0 as $x_0 = \mathcal{G}_\theta(x_T)$.

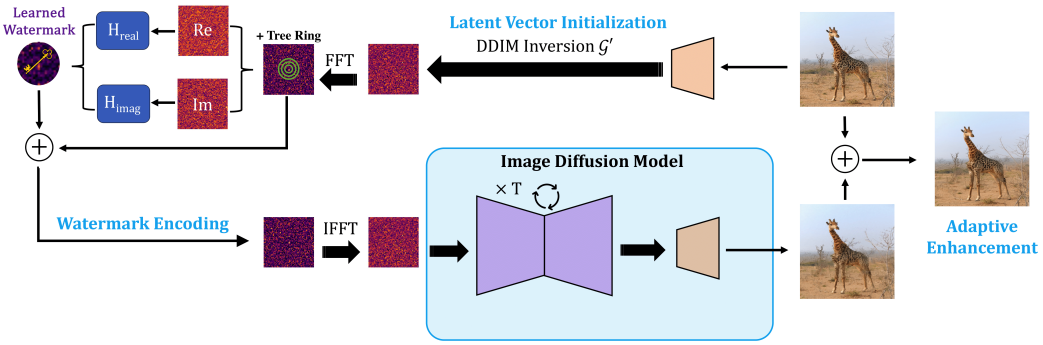
However, given the learned model $\epsilon_\theta(x_t)$, it is also possible to move in the opposite direction. Starting from an image x_0, Dhariwal & Nichol (2021) describe an inverse process that retrieves an initial noise latent Z_T, which maps to an image $\hat{x}_0$ close to x_0 through DDIM, where $\mathcal{G}(x_T) \approx x_0$. This inverse process depends on the assumption that $x_{t-1} \approx \tilde{x}_0^t - x_t$. Therefore, from $x_t \to x_{t+1}$, we follow:

$$x_{t+1} = \sqrt{\bar{\alpha}_{t+1}}\tilde{x}_0^t + \sqrt{1-\bar{\alpha}_{t+1}}\epsilon_\theta(x_t). \tag{5}$$

We denote this whole inversion process from a starting real image x_0 to x_T as $x_T = \mathcal{G}'(x_0)$. Furthermore we treat the initial latent vector x_T as Z_T.

3 Approach

WaterFlow is a novel zero shot watermarking scheme that leverages invertible transformations as well as pretrained stable diffusion models to provide *robust, fast, and high fidelity* watermarking. Our model implants a learned watermark in the Fourier transformed latent space of an image, which is detectable by taking the image back to latent space. We provide an overview in Figure 1.



(a) Generation Phase of WATERFLOW

(b) Detection Phase of WATERFLOW

Figure 1: Overview of WATERFLOW. In (a), we show the generation phase, then (b) shows the detection phase.

Algorithm 1 WaterFlow Watermark

Require: Image x_0 and binary mask M
Require: pre-trained LDM $\mathcal{G}$ and inversion $\mathcal{G}'$
Require: T diffusion steps
Require: Trained mapping H_{real} and H_{imag}
Require: Circular Tree-Ring Watermark W_{Tree}
Require: SSIM Threshold s^*

$Z_T = \mathcal{G}'(x_0)$
$\mathcal{F}(Z'_T) = \mathcal{F}(Z_T) \odot (1 - M) + M \odot W_{\text{Tree}}$
$W^* = H_{\text{real}}(\Re(\mathcal{F}(Z'_T))) + j \cdot H_{\text{imag}}(\Im(\mathcal{F}(Z'_T)))$
$\mathcal{F}(Z_{W^*}) = \mathcal{F}(Z'_T)\,[-1, :, :] \odot (1 - M) + M \odot W^*$
$\hat{x}_0 = \mathcal{G}(Z_{W^*})$
Search $\gamma \in [0, 1]$ s.t. SSIM$(\bar{x}_0, x_0) \geq s$
$\bar{x}_0 = \hat{x}_0 + \gamma(x_0 - \hat{x}_0)$
return $\bar{x}_0$

3.1 WATERMARKING PROCEDURE

We provide the general framework for our approach in Algorithm 1. Note that $\mathcal{F}$ is the Fourier Transform and $\odot$ denotes a Hadamard product. Given an original image x_0, our goal is to generate a new image $\hat{x}_0$ that closely resembles x_0 and incorporates a watermark that can be easily detected.

We first apply DDIM inversion yielding a corresponding latent Z_T, where $Z_T = \mathcal{G}'(x_0)$. Then, we initialize our latent with a predefined watermark, specifically the Tree-Ring Watermark (Wen et al., 2023). This involves implanting a watermark W_{Tree} into $\mathcal{F}(Z_T)$, yielding $\mathcal{F}(Z'_T)$ as shown in Equation 6. W_{Tree} is composed of a series of concentric rings where each ring is sampled from the Fourier Transform of an isotropic normal distribution.

$$\mathcal{F}(Z'_T) = \mathcal{F}(Z_T) \odot (1 - M) + M \odot W_{\text{Tree}} \tag{6}$$

We train two models jointly, H_{real} and H_{imag}, to generate a *learned watermark dependent on the input*. Our watermarks are embedded in the Fourier domain of the latent space, requiring us to account for complex values. One neural network processes the real component of the transformed latent representation, while the other operates on the imaginary component. The outputs are then

combined into a single complex-valued watermark shown in Equation 7.

$$W^* = H_{\text{real}}(\Re(\mathcal{F}(Z'_T))) + j \cdot H_{\text{imag}}(\Im(\mathcal{F}(Z'_T))) \tag{7}$$

This watermark is then planted into the Fourier domain of our latent, which offers a number of strong invariances to classic modifications such as blurring, jittering, and translation Wen et al. (2024). There are many previous works which also exploit similar properties of the Fourier domain for watermarking (Pitas, 1998; Solachidis & Pitas, 2001).

Suppose that our latent has dimension channels c, width w, and height h. We define $M \in \{0,1\}^{w \times h}$ as a circular binary mask of radius r used to embed our watermark and $W^* \in \mathbb{C}^{w \times h}$ as the watermark, which is embedded in the last channel of our transformed latent.

$$M(x,y) = \begin{cases} 1 & \text{if } \sqrt{x^2+y^2} \leq r \\ 0 & \text{else} \end{cases} \tag{8}$$

We apply our watermark in the last channel of the Fourier domain of our latent.

$$\mathcal{F}(Z_{W^*}) = \mathcal{F}(Z'_T)\,[-1,:,:] \;\odot\; (1-M) + M \odot\; W^* \tag{9}$$

Equation 9 shows how we implant our learned watermark into the original latent. This is done by masking out the circular region of $\mathcal{F}(Z'_T)$ and replacing it with the contents of W^*.

After embedding the watermark, our goal is to generate a new image. This is achieved by applying the Inverse Fourier Transform (IFT) and subsequently using the resulting latent representation as the initial noise vector for diffusion ($\mathcal{G}$).

To balance watermark detectability with image quality, we adopt the adaptive image enhancement approach from ZoDiac (Zhang et al., 2024). This method involves reintroducing a fraction ($\gamma \in [0,1]$) of the original image to satisfy a predefined Structural Similarity Index (SSIM) threshold (s^*). The enhanced image, denoted as $\bar{x}_0$, is defined as follows:

$$\bar{x}_0 = \gamma(x_0) + (1-\gamma)\hat{x}_0 \tag{10}$$

Note that x_0 is the original image and $\hat{x}_0$ is the watermarked image. We find the minimum γ such that $\text{SSIM}(\bar{x}_0, x_0) \geq s^*$ via binary search.

3.2 Trainable Watermark

In the literature, Wen et al. (2024) uses a fixed circular watermark and Zhang et al. (2024) optimizes the latent itself before watermarking. We combine the motivation for both by creating a learned watermark, which has the flexibility of learning a per-latent concept while only altering a small portion of the latent.

To do so, we create a low parameter and efficient models for H_{real} and H_{imag}. The model architecture that we employ are known as Residual Flows (Chen et al., 2019), a popular invertible generative modeling method. We provide further details for this choice in the Appendix C.4 and section 5.2.

The loss functions that we use to optimize these mappings are as follows:

$$\mathcal{L} = \lambda_2\mathcal{L}_2(x_0,\hat{x}_0) + \lambda_s\mathcal{L}_s(x_0,\hat{x}_0) + \lambda_p\mathcal{L}_p(x_0,\hat{x}_0) + \lambda_n\mathcal{L}_n(x_0,W^*,M) \tag{11}$$

$\mathcal{L}_2$ represents the mean squared error between the generated and ground truth image, $\mathcal{L}_s$ is SSIM loss (Zhao et al., 2017), and $\mathcal{L}_p$ is the VGG-perceptual loss (Johnson et al., 2016). λ_2, λ_s, and λ_p represent the respective loss weights.

The last loss $\mathcal{L}_n$ is the negative mean squared error between the watermarked region of the original image and the newly derived watermark (with corresponding loss weight λ_n). We want to encourage our models to learn a watermark that maximizes the distance between the two, creating a fundamentally more detectable watermark. More formally, we define

$$\mathcal{L}_n = -\frac{1}{w \cdot h}\sum_{i=1}^{w}\sum_{j=1}^{h}\Big([\mathcal{F}(x_0)\odot M]_{i,j} - [W^* \odot M]_{i,j}\Big)^2 \tag{12}$$

Our optimization aligns precisely with the detection method shown in Equation 14.

3.3 WATERMARK DETECTION

In the detection phase, we want to verify the presence of a watermark in a given image x_0 with high certainty. Since our watermark was placed in the latent space, we take the image back to this space, i.e., we let $y = \mathcal{F}(\mathcal{G}'(x_0))[-1, :, :]$. Note that we place the watermark in the last channel of the latent. We use statistical tests to assess the likelihood that our watermark W^* is embedded in y by random chance. Additionally, a p-value enables the setting of a task-specific threshold, which can be adjusted to control the detection difficulty.

We know that the implanted watermark always follows a Gaussian distribution. Hence, we define the following as our null Hypothesis:

$$H_0 : y \sim \mathcal{N}(0, \sigma^2 I_{\mathbb{C}}) \tag{13}$$

We estimate σ^2 for each image by computing $\sigma^2 = \frac{1}{\sum M} \sum M \odot y$. We then define a score η as follows:

$$\eta = \frac{1}{\sigma^2} \sum (M \odot W^* - M \odot y)^2 \tag{14}$$

This scores quantitatively measures the difference between the watermark and the watermark portion of y. Furthermore, because y is normal, we know that our score η follows a non-central chi-squared distribution (PATNAIK, 1949). This distribution has $q = \sum M$ degrees of freedom and a non-centrality parameter $\lambda = \frac{1}{\sigma^2} \sum (M \odot W^*)^2$.

If η is sufficiently small, then we can reasonably conclude that our image was not watermarked by random chance. To do this, we employ the cumulative distribution function of the non-central χ^2 distribution:

$$p = Pr(\chi^2_{q,\lambda} \leq \eta \mid H_0) \tag{15}$$

Watermarked images will exhibit extremely low p-value while non-watermarked images will have much higher p-values. We define the detection probability as $1 - p$ for our results.

4 EXPERIMENTS

In this section, we detail the datasets, settings used by our approach, the baselines used for comparison, robustness, and runtime performance.

Table 1: We display the Watermark Detection rate after the watermarked images go through a series of perturbations or attacks. After the image is transformed we would hope that the mark persists.

		Post-Attack												
Dataset	**Method**	**Brightness**	**Contrast**	**JPEG**	**Rotation**	**G-Noise**	**G-Blur**	**BM3D**	**Bmshj18**	**Cheng20**	**Zhao23**	**All**	**All w/o Rotation**	**Overall Avg.**
MS-COCO	DwtDct	0.000	0.000	0.000	0.000	0.630	0.230	0.000	0.000	0.000	0.000	0.000	0.000	0.072
	DwtDctSvd	0.080	0.080	0.750	0.000	0.480	1.000	1.000	0.480	0.030	0.160	0.000	0.000	0.338
	RivaGAN	1.000	1.000	1.000	0.000	1.000	1.000	1.000	0.010	0.010	0.050	0.000	0.000	0.506
	SSL	1.000	1.000	0.250	1.000	0.330	1.000	0.030	0.010	0.010	0.040	0.000	0.000	0.389
	ZoDiac	0.960	0.970	0.930	0.430	0.950	0.970	0.950	0.910	0.920	0.880	0.150	0.420	0.787
	Tree-Ring	0.730	0.730	0.570	0.250	0.630	0.720	0.640	0.620	0.540	0.510	0.130	0.310	0.532
	WF (Ours)	1.000	1.000	1.000	0.810	1.000	1.000	1.000	1.000	1.000	0.990	0.880	0.990	**0.973**
DIFFDB	DwtDct	0.000	0.000	0.000	0.000	0.490	0.410	0.000	0.000	0.000	0.000	0.000	0.000	0.075
	DwtDctSvd	0.070	0.080	0.880	0.000	0.980	1.000	0.630	0.020	0.060	0.110	0.000	0.000	0.319
	RivaGAN	0.970	0.970	0.930	0.000	0.990	0.990	0.910	0.020	0.010	0.060	0.000	0.000	0.488
	SSL	0.990	1.000	0.420	0.980	0.270	1.000	0.050	0.000	0.020	0.050	0.000	0.000	0.398
	ZoDiac	0.970	0.980	0.960	0.460	0.950	0.980	0.950	0.930	0.920	0.860	0.110	0.500	0.798
	Tree-Ring	0.810	0.820	0.700	0.300	0.710	0.820	0.750	0.670	0.690	0.530	0.100	0.350	0.604
	WF (Ours)	0.990	0.990	0.990	0.440	0.990	0.990	0.990	0.990	0.980	0.950	0.300	0.870	**0.872**
WIKIART	DwtDct	0.000	0.000	0.000	0.000	0.530	0.300	0.000	0.000	0.000	0.000	0.000	0.000	0.069
	DwtDctSvd	0.120	0.110	0.800	0.000	1.000	1.000	0.580	0.040	0.060	0.120	0.000	0.000	0.319
	RivaGAN	0.990	0.990	0.980	0.000	0.990	0.990	0.980	0.020	0.020	0.070	0.000	0.000	0.503
	SSL	0.990	1.000	0.260	0.970	0.530	1.000	0.080	0.000	0.020	0.010	0.000	0.000	0.405
	ZoDiac	0.960	0.970	0.900	0.350	0.950	0.950	0.940	0.830	0.810	0.800	0.110	0.430	0.750
	Tree-Ring	0.690	0.720	0.660	0.190	0.600	0.710	0.620	0.520	0.510	0.500	0.080	0.180	0.498
	WF (Ours)	0.980	0.980	0.990	0.660	0.990	0.980	0.980	0.970	0.990	0.950	0.760	0.900	**0.928**

4.1 SET UP

In our experiments, we utilize three datasets to evaluate our approach. First, we select 100 real images randomly sampled from the MS-COCO dataset Lin et al. (2014), a widely-used benchmark for image recognition and segmentation tasks. To represent AI-generated images, we use 100 samples

from DiffusionDB Wang et al. (2022), a dataset containing images generated through actual user interactions with Stable Diffusion, including diverse prompts and hyperparameter configurations. Finally, we include 100 images from the WikiArt dataset Phillips & Mackintosh (2011), a curated collection of artworks that spans various artistic styles and periods, sourced from WikiArt.org.

For baseline comparisons, we evaluate against six key methods: ZoDiac Zhang et al. (2024), Tree-Ring Wen et al. (2023), DwtDct Cox et al. (2007), and DwtDctSvd Navas et al. (2008), Riva-Gan Zhang et al. (2019c), and SSL Fernandez et al. (2023) on the same set of images.

For information about the metrics we used for computing image quality, robustness, and speed, please refer to Appendix D. Most metrics are typical except the Watermark detection rate. The returned p-value of an image we consider an image watermarked if the detection probability is greater than some threshold p^*.

4.2 Watermarking Attacks

To benchmark the robustness of our watermarking method, we evaluate its performance under common data augmentations and perturbations. We utilize the following attacks in our assessment: **Brightness** and **Contrast**: with a factor of 0.5, **JPEG**: compression with a quality setting of 50, **Rotation**: by 90 degrees, **G-Noise**: Addition of Gaussian noise with std of 0.05, **G-Blur**: Gaussian blur with kernel size 5 and std 1, **BM3D**: Denoising algorithm with a std of 0.1, **Bmshj18** and **Cheng20**: Two Variational AutoEncoder (VAE) based image compression models, both with compression factors of 3 Ballé et al. (2018); Cheng et al. (2020), **Zhao23**: Stable diffusion-based image regeneration model, with 60 denoising steps Zhao et al. (2023a), **All**: Combination of all the attacks, and **All w/o Rotation**: Combination of all the attacks without rotation

Table 2: We show the average time taken (in seconds) to evaluate a single image with three approaches. We also report the variance.

ZoDiac	Tree-Ring	WF (Ours)
517.68 ± 6.00	4.85 ± 0.00	6.43 ± 0.00

Table 3: We display the AUC after the watermarked images go through a series of perturbations or attacks. This is done by treating the base image as negative and the watermarked as positive. After the image is transformed we would hope that the mark persists. First block is MS-COCO, second is DiffusionDB, and third WikiArt. We bold the highest average attack AUC.

		Post-Attack												
Dataset	**Method**	**Brightness**	**Contrast**	**JPEG**	**Rotation**	**G-Noise**	**G-Blur**	**BM3D**	**Bmshj18**	**Cheng20**	**Zhao23**	**All**	**All w/o Rotation**	**Overall Avg.**
MS-COCO	DwtDct	0.500	0.490	0.550	0.415	0.922	0.790	0.549	0.517	0.507	0.496	0.499	0.505	0.562
	DwtDctSvd	0.480	0.481	0.993	0.633	1.000	1.000	0.960	0.804	0.741	0.773	0.464	0.494	0.735
	RivaGan	1.000	1.000	1.000	0.502	1.000	1.000	1.000	0.726	0.750	0.868	0.523	0.509	0.823
	SSL	1.000	1.000	0.852	1.000	0.881	1.000	0.698	0.608	0.662	0.711	0.485	0.496	0.783
	ZoDiac	0.991	0.996	0.978	0.754	0.986	0.991	0.989	0.975	0.976	0.961	0.584	0.765	0.912
	Tree-Ring	0.929	0.929	0.851	0.674	0.869	0.934	0.914	0.868	0.854	0.848	0.595	0.716	0.832
	WF (Ours)	0.999	0.999	0.999	0.913	0.999	1.000	0.999	0.999	0.999	0.998	0.940	0.985	**0.986**
DIFFDB	DwtDct	0.495	0.509	0.546	0.387	0.902	0.831	0.565	0.488	0.487	0.491	0.523	0.515	0.562
	DwtDctSvd	0.377	0.387	0.989	0.671	0.984	1.000	0.976	0.825	0.852	0.736	0.492	0.524	0.734
	RivaGan	0.998	0.997	0.994	0.468	0.999	0.999	0.991	0.703	0.670	0.742	0.494	0.574	0.802
	SSL	1.000	1.000	0.852	0.995	0.821	1.000	0.760	0.619	0.584	0.665	0.531	0.502	0.777
	ZoDiac	0.994	0.994	0.989	0.800	0.989	0.996	0.991	0.988	0.984	0.960	0.622	0.836	0.929
	Tree-Ring	0.957	0.951	0.932	0.703	0.927	0.949	0.922	0.912	0.900	0.866	0.553	0.757	0.861
	WF (Ours)	1.000	1.000	1.000	0.903	0.999	1.000	1.000	1.000	0.999	0.999	0.879	0.967	**0.978**
WIKIART	DwtDct	0.502	0.509	0.567	0.373	0.883	0.799	0.538	0.505	0.519	0.503	0.502	0.516	0.560
	DwtDctSvd	0.525	0.526	0.993	0.768	1.000	1.000	0.964	0.845	0.843	0.744	0.517	0.502	0.769
	RivaGan	1.000	1.000	1.000	0.479	1.000	1.000	1.000	0.757	0.673	0.830	0.458	0.495	0.808
	SSL	0.999	1.000	0.856	0.997	0.916	1.000	0.679	0.588	0.618	0.637	0.455	0.476	0.768
	ZoDiac	0.991	0.991	0.981	0.732	0.988	0.993	0.984	0.964	0.960	0.956	0.572	0.812	0.910
	Tree-Ring	0.934	0.930	0.905	0.695	0.917	0.938	0.915	0.873	0.874	0.861	0.546	0.728	0.843
	WF (Ours)	0.997	0.998	0.999	0.937	1.000	0.999	0.999	0.996	0.996	0.995	0.896	0.986	**0.983**

4.3 Robustness Results and Discussion

We present our robustness results in Table 1 and Table 3. Table 5 supplements Table 3 and is located in Appendix C.1. We observe that *WaterFlow has the highest overall average WDR, AUC, and TPR1%FPR across all datasets*.

Against traditional attacks such as brightness, contrast, and etc., WaterFlow achieves near-perfect watermark detection accuracy, which is competitive with existing state-of-the-art.

On rotation, a well documented difficult attack in watermarking, WaterFlow achieves AUC values as high as 0.937, which is the second best method (shown in Table 3). SSL's performance advantage

Table 4: Image quality experiments showing PSNR, SSIM, and pre-attack detection probability (watermark detection rate) as a representation of fidelity as well as how likely we are to detect whether the image is indeed marked. We bold the highest value.

Method	MS-COCO				DiffusionDB				WikiArt			
	PSNR ↑	SSIM ↑	Det. Prob ↑	LPIPS ↓	PSNR ↑	SSIM ↑	Det. Prob ↑	LPIPS ↓	PSNR ↑	SSIM ↑	Det. Prob ↑	LPIPS ↓
DwtDct	39.46	0.97	0.869	**0.03**	38.33	0.970	0.848	**0.03**	38.12	0.98	0.848	**0.02**
DwtDctSvd	39.42	**0.98**	**1.000**	**0.03**	38.18	**0.98**	**1.000**	**0.03**	38.10	0.98	**1.000**	0.04
RivaGAN	40.63	**0.98**	**1.000**	0.07	40.48	**0.98**	0.988	0.07	40.41	**0.99**	0.997	0.08
SSL	**41.78**	**0.98**	**1.000**	0.09	**41.81**	**0.98**	**1.000**	0.08	**41.78**	**0.99**	**1.000**	0.07
ZoDiac	28.61	0.92	0.992	0.13	28.65	0.92	0.995	0.11	28.93	0.92	0.990	0.10
Tree-Ring	25.71	0.92	0.926	0.13	25.74	0.92	0.951	0.11	25.96	0.92	0.910	0.12
WF (Ours)	27.74	0.95	**1.000**	0.12	27.39	0.95	0.998	0.09	26.94	0.95	0.994	0.10

can be attributed to its use of rotation augmentation during training, which our approach does not incorporate. By foregoing augmentation, we inherently improve robustness without relying on prior knowledge of attack characteristics.

More recent attacks, such as Bmshj18, Cheng20, and Zhao23, primarily leverage generative models to regenerate images, effectively removing embedded marks. older methods like DwtDct, DwtDctSvd, RivaGan, and SSL fail to defend against these attacks Zhao et al. (2023a). WaterFlow provides the highest WDR and AUC/TPR1%FPR across all three of these generative attacks. We reason that the DDIM inversion process offers significant protection, as modifications in the image space remain closely aligned in the latent space. This explains the resilience shown by Tree-Ring, ZoDiac, and WaterFlow.

In the most challenging scenarios—All and All w/o Rotation—WaterFlow exhibits exceptional robustness, far surpassing all other methods. In contrast, ZoDiac and other baselines offer little to no protection. WaterFlow achieves near-perfect AUC in All w/o Rotation and maintains AUC greater than 0.940 across all datasets. Its WDR reaches 0.990, and in the All attack setting, it consistently achieves AUC greater than 0.800. These results mark a significant improvement over existing baselines, establishing WaterFlow as the first method to effectively defend against even the most challenging combination attacks.

The key to this improvement lies in our newly introduced loss term, $\mathcal{L}_n$, which directly optimizes for robustness. As the loss magnitude increases, the watermark inherently becomes more detectable. In Equation 12, we see that the loss term encourages greater separation between $\mathcal{F}(x_0) \odot M$ and $W^* \odot M$, making the watermark more perceptible and consequently more resilient. A similar intuition underlies the zero-watermark baseline (inserting an all-zero patch) proposed in the Tree-Ring paper Wen et al. (2023). Our approach enables the watermark to be learned dynamically without imposing a predefined structure, resulting in greater flexibility. This adaptability allows the mapping to optimize the trade-off between detectability and image quality on a per-latent basis.

4.4 Watermarking Speed Results and Discussion

We evaluate the time required to watermark a single image within diffusion-based watermarking methods. As shown in Table 2, our approach is slightly slower than Tree-Ring but almost 90 times faster than ZoDiac. This speedup stems from a key difference: while ZoDiac optimizes each latent individually, our method learns a universal mapping, relying solely on model inference and avoiding costly gradient descent. Tree-Ring is faster as it omits watermark creation and embedding. Efficient inference is crucial, as ZoDiac's slow performance hinders real-world usability.

4.5 Image Quality Results and Discussion

We present our results on image quality in Table 4. On all three datasets, WaterFlow shows competitive and even better results compared to the most relevant baselines, i.e., Tree-Ring and ZoDiac. WaterFlow does better than Tree-Ring on every perceptual metric and better than ZoDiac on SSIM and LPIPS. This is partly due to our watermark's resilience to modifications in image space, enabling us to restore more of the original image with minimal impact on detectability when postprocessing. (We can view adaptive enhancement as a variation of some adversarial attack.) In Appendix C.6, we present an ablation study exploring SSIM threshold variations (for adaptive enhancement) to improve image quality while maintaining strong robustness.

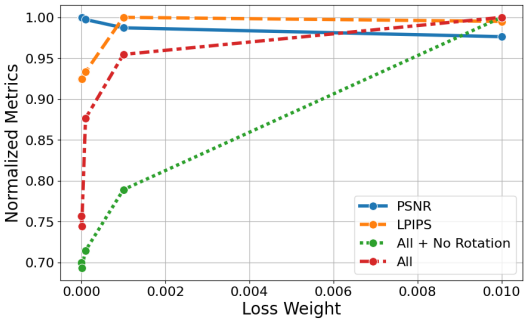


Figure 2: We compare 5 different loss weights for λ_n, i.e., $10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}$. We evaluate on 4 different metrics, PSNR, LPIPS, rotation, all of the attacks + no rotation, and then finally all the attacks. Note that PSNR and LPIPS are normalized.

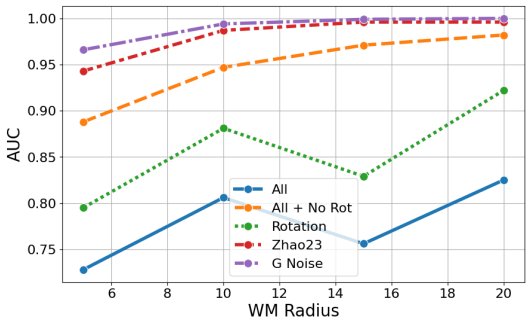


Figure 3: Ablation on watermark radius. We show AUC across a wide variety of different attacks and perturbations using radius 5, 10, 15, 20.

5 Ablation Study

5.1 The loss weight parameters, λ_n

We need to account for the loss associated with the newly proposed loss term. This requires careful balancing to ensure that the negative MSE does not overwhelm the optimization process.

Our results can be seen in Figure 2 with more details in Appendix C.2. In total, we tried $\lambda_n \in \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. Our results clearly indicate a direct trade-off between perception metrics and detectability/robustness. This is because a larger weight encourages our optimization to prioritize the watermark's detectability (L2 loss) over the perceptual quality of the image.

5.2 Model Architecture

An additional consideration is the architecture we use to parameterize the learned mappings. We provided full results in the Appendix C.4. In our experiments, we look at UNet-Based Ronneberger et al. (2015), MLP, and Residual Net Chen et al. (2019).

While UNet and MLP slightly outperform on perceptual metrics, they lack robustness. UNet tends to make the learned watermark resemble the ground truth patch, increasing false positives. MLP fares slightly better but still struggles. We chose FlowNets for their ability to map simple distributions to complex ones while preserving volume and retaining information. Given the Gaussian latent space, our goal was to transform it into a more complex, watermarked distribution. Their invertible architecture also helps avoid local optima, enhancing diversity in the learned watermark.

5.3 Watermark Radius

We provide our full results in Appendix C.3. We explore the effects of modulating the radius on overall image quality and robustness. We evaluate our approach with radius $5, 10, 15, 20$.

As shown in Figure 3, increasing the radius of the watermark generally leads to better robustness. This is because a larger watermark covers more area, resulting in greater deviation from the unwatermarked region. However, this comes at the expense of image quality. Our results show a slight decrease in PSNR as the radius increases: a watermark radius of 20 yields a PSNR of 25.10, compared to 25.49 for a radius of 5. Since a smaller watermark radius preserves more of the original latent space, we expect better image quality.

6 Conclusion

This work introduces WaterFlow, a fast, robust, and high-quality watermarking method leveraging latent diffusion models. WaterFlow encodes an image into latent space using a pretrained diffusion model, embedding a latent-dependent watermark for high detectability with minimal disruption. It achieves state-of-the-art robustness across benchmarks (DiffusionDB, MS-COCO, WikiArt) while boasting extremely quick watermarking speeds. Notably, WaterFlow is the first method to defend against complex combination attacks, overcoming a key limitation in field.

REFERENCES

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

Walter Bender, Daniel Gruhl, Norishige Morimoto, and Anthony Lu. Techniques for data hiding. *IBM systems journal*, 35(3.4):313–336, 1996.

A.G. Bors and I. Pitas. Image watermarking using dct domain constraints. In *Proceedings of 3rd IEEE International Conference on Image Processing*, volume 3, pp. 231–234 vol.3, 1996.

Ricky TQ Chen, Jens Behrmann, David K Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 32, 2019.

Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7939–7948, 2020.

Ingemar Cox, Matthew Miller, Jeffrey Bloom, and Chris Honsinger. Digital watermarking. *Journal of Electronic Imaging*, 11(3):414–414, 2002.

Ingemar Cox, Matthew Miller, Jeffrey Bloom, Jessica Fridrich, and Ton Kalker. *Digital Watermarking and Steganography*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2 edition, 2007. ISBN 9780080555805.

Yingqian Cui, Jie Ren, Han Xu, Pengfei He, Hui Liu, Lichao Sun, and Jiliang Tang. Diffusionshield: A watermark for copyright protection against generative diffusion models. *arXiv preprint arXiv:2306.04642*, 2023.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. The stable signature: Rooting watermarks in latent diffusion models. *arXiv preprint arXiv:2303.15435*, 2023.

Giorgio Franceschelli and Mirco Musolesi. Copyright in generative deep learning. *Data & Policy*, 4:e17, 2022.

Jamie Hayes and George Danezis. Generating steganographic images via adversarial training. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Kun Hu, Mingpei Wang, Xiaohui Ma, Jia Chen, Xiaochao Wang, and Xingjun Wang. Learning-based image steganography and watermarking: A survey. *Expert Systems with Applications*, pp. 123715, 2024.

Jiangtao Huang, Ting Luo, Li Li, Gaobo Yang, Haiyong Xu, and Chin-Chen Chang. Arwgan: Attention-guided robust image watermarking model based on gan. *IEEE Transactions on Instrumentation and Measurement*, 72:1–17, 2023.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pp. 694–711. Springer, 2016.

Deepa Kundur and Dimitrios Hatzinakos. Digital watermarking using multiresolution wavelet decomposition. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 5, pp. 2969–2972. IEEE, 1998.

Liangqi Lei, Keke Gai, Jing Yu, and Liehuang Zhu. Diffusetrace: A transparent and flexible watermarking scheme for latent diffusion model. *arXiv preprint arXiv:2405.02696*, 2024.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Xiyang Luo, Ruohan Zhan, Huiwen Chang, Feng Yang, and Peyman Milanfar. Distortion agnostic deep watermarking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13548–13557, 2020.

Rui Ma, Mengxi Guo, Yi Hou, Fan Yang, Yuan Li, Huizhu Jia, and Xiaodong Xie. Towards blind watermarking: Combining invertible and non-invertible mechanisms. In *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 1532–1542, 2022.

Rui Min, Sen Li, Hongyang Chen, and Minhao Cheng. A watermark-conditioned diffusion model for ip protection, 2024.

K. A. Navas, Mathews Cheriyan Ajay, M. Lekshmi, Tampy S. Archana, and M. Sasikumar. Dwt-dct-svd based watermarking. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, pp. 271–274, 2008. doi: 10.1109/COMSWA.2008.4554423.

P. B. PATNAIK. THE NON-CENTRAL X2 AND F-DISTRIBUTIONS AND THEIR APPLICATIONS†. *Biometrika*, 36(1-2):202–232, 06 1949. ISSN 0006-3444. doi: 10.1093/biomet/36.1-2.202. URL https://doi.org/10.1093/biomet/36.1-2.202.

Fred Phillips and Brandy Mackintosh. Wiki art gallery, inc.: A case for critical thinking. *Issues in Accounting Education*, 26(3):593–608, 2011.

I. Pitas. A method for watermark casting on digital image. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(6):775–780, 1998. doi: 10.1109/76.728421.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241. Springer, 2015.

V. Solachidis and L. Pitas. Circularly symmetric watermark embedding in 2-d dft domain. *IEEE Transactions on Image Processing*, 10(11):1741–1753, 2001. doi: 10.1109/83.967401.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Yuqi Tan, Yuang Peng, Hao Fang, Bin Chen, and Shu-Tao Xia. Waterdiff: Perceptual image watermarks via diffusion model. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3250–3254, 2024. doi: 10.1109/ICASSP48485.2024.10447095.

Matthew Tancik, Ben Mildenhall, and Ren Ng. Stegastamp: Invisible hyperlinks in physical photographs. *CoRR*, abs/1904.05343, 2019.

Matthieu Urvoy, Dalila Goudia, and Florent Autrusseau. Perceptual dft watermarking with improved detection and robustness to geometrical distortions. *IEEE Transactions on Information Forensics and Security*, 9(7):1108–1119, 2014.

Zhenting Wang, Chen Chen, Yuchen Liu, Lingjuan Lyu, Dimitris Metaxas, and Shiqing Ma. How to detect unauthorized data usages in text-to-image diffusion models. *arXiv preprint arXiv:2307.03108*, 2023.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

Zijie J Wang, Evan Montoya, David Munechika, Haoyang Yang, Benjamin Hoover, and Duen Horng Chau. Diffusiondb: A large-scale prompt gallery dataset for text-to-image generative models. *arXiv preprint arXiv:2210.14896*, 2022.

Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-ring watermarks: Fingerprints for diffusion images that are invisible and robust. *arXiv preprint arXiv:2305.20030*, 2023.

Yuxin Wen, John Kirchenbauer, Jonas Geiping, and Tom Goldstein. Tree-rings watermarks: Invisible fingerprints for diffusion images. *Advances in Neural Information Processing Systems*, 36, 2024.

Raymond B Wolfgang and Edward J Delp. A watermark for digital images. In *Proceedings of 3rd IEEE International Conference on Image Processing*, volume 3, pp. 219–222. IEEE, 1996.

Xiang-Gen Xia, Charles G. Boncelet, and Gonzalo R. Arce. Wavelet transform based watermark for digital images. *Opt. Express*, 3(12):497–511, Dec 1998.

Zijin Yang, Kai Zeng, Kejiang Chen, Han Fang, Weiming Zhang, and Nenghai Yu. Gaussian shading: Provable performance-lossless image watermarking for diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12162–12171, 2024.

Kevin Alex Zhang, Alfredo Cuesta-Infante, Lei Xu, and Kalyan Veeramachaneni. Steganogan: High capacity image steganography with gans. *CoRR*, abs/1901.03892, 2019a.

Kevin Alex Zhang, Lei Xu, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Robust invisible video watermarking with attention. *CoRR*, abs/1909.01285, 2019b.

Kevin Alex Zhang, Lei Xu, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Robust invisible video watermarking with attention. *arXiv preprint arXiv:1909.01285*, 2019c.

Lijun Zhang, Xiao Liu, Antoni Viros Martin, Cindy Xiong Bearfield, Yuriy Brun, and Hui Guan. Robust image watermarking using stable diffusion. *arXiv preprint arXiv:2401.04247*, 2024.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586–595, 2018.

Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, 3(1):47–57, 2017. doi: 10.1109/TCI.2016.2644865.

Xuandong Zhao, Kexun Zhang, Yu-Xiang Wang, and Lei Li. Generative autoencoders as watermark attackers: Analyses of vulnerabilities and threats. *arXiv preprint arXiv:2306.01953*, 2023a.

Yunqing Zhao, Tianyu Pang, Chao Du, Xiao Yang, Ngai-Man Cheung, and Min Lin. A recipe for watermarking diffusion models. *arXiv preprint arXiv:2303.10137*, 2023b.

Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. Hidden: Hiding data with deep networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

A TRAINING SET

Our training set consists of 300 samples per dataset. All of our datasets are pulled from publically available huggingface APIs.

B EXPERIMENT DETAILS

All of our training is done on a single NVIDIA A6000 and A5000 GPUs and Intel(R) Xeon(R) Gold 6342 CPU @ 2.80GHz. Most of the training was able to complete in less than a couple of hours.

B.1 HYPERPARAMETERS

We train for a maximum of 15 epochs, Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and learning rate 0.001. All our experiments use a batch size of 2. We use 50 denoising steps for our diffusion model. We also use a SSIM threshold of 0.95 for adaptive enhancement on our main results across all datasets.

For the loss function, we use $\lambda_n = 10^{-2}$, $\lambda_2 = 10.0$, $\lambda_s = 0.1, \lambda_p = 1.0$.

B.2 CHECKPOINT SELECTION

We save a checkpoint every 50 steps and take the checkpoint with the lowest loss.

B.3 BASELINES

For ZoDiac, we adhere to the parameter settings specified in its original paper Zhang et al. (2024), including an SSIM threshold of 0.92 and 100 epochs. For Tree-Ring, we adapt the original method—designed for watermarking only diffusion-generated images—to support watermarking arbitrary images. This adaptation involves performing DDIM inversion on an input image, embedding the Tree-Ring watermark into the latent space, and then regenerating the image. We also use adaptive enhancement with Tree-Rings. For DwtDct, DwtDctSvd, SSL, and RivaGAN, we embed a 32-bit message and set a watermark detection threshold of 24/32 correctly predicted bits. Each of these methods is executed using its default parameters as provided in their respective implementations Cox et al. (2007); Navas et al. (2008); Zhang et al. (2019c); Fernandez et al. (2023).

B.4 MODELS

We present the model architecture for the MLP as coded in PyTorch:

```
torch.nn.Sequential(
        torch.nn.Flatten(),
        torch.nn.Linear(4*64*64, 64),
        torch.nn.ReLU(),
        torch.nn.Linear(64, 4*64*64),
        torch.nn.Unflatten(1, (4, 64, 64))
    )
```

As well as the model architecture for WaterFlow implemented in the popular normflows library:

```
K = 2

latent_size = (4, 64, 64)
hidden_units = 64
hidden_layers = 3

flows = []
for i in range(K):
    net = nf.nets.LipschitzCNN([4] +
```

```
    [hidden_units]*(hidden_layers - 1)
    + [4], [3, 1, 3], init_zeros=True,
    lipschitz_const=0.9)
    flows += [nf.flows.Residual(net)]
```

We present the model architecture for the Unet:

```
class UNet(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(UNet, self).__init__()

        self.encoder1 = DoubleConv(in_channels, 8)
        self.encoder2 = DoubleConv(8, 16)
        self.encoder3 = DoubleConv(16, 32)

        self.pool = nn.MaxPool2d(2)

        self.bottleneck = DoubleConv(32, 64)

        self.upconv3 = nn.ConvTranspose2d(64, 32,
        kernel_size=2, stride=2)
        self.decoder3 = DoubleConv(64, 32)

        self.upconv2 = nn.ConvTranspose2d(32, 16,
        kernel_size=2, stride=2)
        self.decoder2 = DoubleConv(32, 16)

        self.upconv1 = nn.ConvTranspose2d(16, 8,
        kernel_size=2, stride=2)
        self.decoder1 = DoubleConv(16, 8)

        self.final_conv = nn.Conv2d(8, out_channels,
        kernel_size=1)

    def forward(self, x):
        # Encoder
        e1 = self.encoder1(x)
        e2 = self.encoder2(self.pool(e1))
        e3 = self.encoder3(self.pool(e2))

        # Bottleneck
        b = self.bottleneck(self.pool(e3))

        d3 = self.upconv3(b)
        d3 = torch.cat((d3, e3), dim=1)
        d3 = self.decoder3(d3)

        d2 = self.upconv2(d3)
        d2 = torch.cat((d2, e2), dim=1)
        d2 = self.decoder2(d2)

        d1 = self.upconv1(d2)
        d1 = torch.cat((d1, e1), dim=1)
        d1 = self.decoder1(d1)

        out = self.final_conv(d1)
        return out
```

C ADDITIONAL RESULTS

C.1 MAIN EXPERIMENT

Table 5: We display the AUC / TPR@1%FPR after the watermarked images go through a series of perturbations or attacks. After the image is transformed we would hope that the mark persists. First block is MS-COCO, second is DiffusionDB, and third WikiArt. We bold the highest average value. Create two tables seperate AUC/TPR%.

Method	Post-Attack												
	Brightness	Contrast	JPEG	Rotation	G-Noise	G-Blur	BM3D	Bmshj18	Cheng20	Zhao23	All	All w/o Rot.	Avg.
DwtDct	0.500/0.010	0.499/0.010	0.550/0.010	0.415/0.000	0.922/0.750	0.790/0.350	0.549/0.000	0.517/0.000	0.507/0.000	0.496/0.010	0.499/0.000	0.505/0.000	0.562/0.095
DwtDctSvd	0.480/0.130	0.481/0.150	0.993/0.900	0.633/0.000	1.000/1.000	1.000/1.000	0.960/0.610	0.804/0.120	0.741/0.100	0.773/0.300	0.464/0.020	0.494/0.010	0.735/0.362
RivaGan	1.000/1.000	1.000/1.000	1.000/1.000	0.502/0.000	1.000/1.000	1.000/1.000	1.000/1.000	0.726/0.100	0.750/0.050	0.868/0.050	0.523/0.010	0.509/0.010	0.823/0.518
SSL	1.000/1.000	1.000/1.000	0.852/0.380	1.000/1.000	0.881/0.420	1.000/1.000	0.698/0.150	0.608/0.010	0.662/0.080	0.711/0.140	0.485/0.000	0.496/0.000	0.783/0.432
ZoDiac	0.991/0.900	0.996/0.960	0.978/0.890	0.754/0.310	0.986/0.900	0.991/0.950	0.989/0.930	0.975/0.790	0.976/0.850	0.961/0.630	0.584/0.040	0.765/0.160	0.912/0.692
Tree-Ring	0.929/0.460	0.929/0.690	0.851/0.370	0.674/0.070	0.869/0.360	0.934/0.540	0.914/0.510	0.868/0.220	0.854/0.380	0.848/0.370	0.595/0.040	0.716/0.110	0.832/0.343
WF (Ours)	1.000/0.990	1.000/0.960	1.000/0.990	0.918/0.380	1.000/0.990	1.000/1.000	0.999/0.930	1.000/0.970	0.999/0.940	0.999/0.990	0.944/0.310	0.994/0.930	**0.988/0.865**
DwtDct	0.495/0.020	0.509/0.020	0.546/0.000	0.387/0.000	0.902/0.600	0.831/0.500	0.565/0.020	0.488/0.000	0.487/0.000	0.491/0.000	0.523/0.000	0.515/0.000	0.562/0.097
DwtDctSvd	0.377/0.170	0.387/0.150	0.989/0.950	0.671/0.000	0.984/0.980	1.000/1.000	0.976/0.630	0.825/0.120	0.852/0.180	0.736/0.190	0.492/0.000	0.524/0.000	0.734/0.364
RivaGan	0.998/0.990	0.997/0.970	0.994/0.990	0.468/0.000	0.999/0.990	0.999/0.990	0.991/0.950	0.703/0.060	0.670/0.110	0.742/0.060	0.494/0.000	0.574/0.000	0.802/0.509
SSL	1.000/0.990	1.000/1.000	0.852/0.460	0.995/0.980	0.821/0.430	1.000/1.000	0.760/0.130	0.619/0.030	0.584/0.040	0.665/0.050	0.531/0.010	0.502/0.020	0.777/0.428
ZoDiac	0.994/0.960	0.994/0.950	0.989/0.880	0.800/0.230	0.989/0.940	0.996/0.940	0.991/0.940	0.988/0.840	0.984/0.870	0.960/0.740	0.622/0.050	0.836/0.140	0.929/0.707
Tree-Ring	0.957/0.770	0.951/0.730	0.932/0.550	0.703/0.180	0.927/0.670	0.949/0.570	0.922/0.560	0.912/0.630	0.900/0.440	0.866/0.420	0.553/0.080	0.757/0.190	0.861/0.483
WF (Ours)	1.000/0.990	1.000/0.990	1.000/0.990	0.903/0.480	0.999/0.960	1.000/0.990	1.000/0.990	1.000/0.990	0.999/0.980	0.985/0.950	0.879/0.500	0.967/0.810	**0.978/0.885**
DwtDct	0.502/0.020	0.509/0.020	0.567/0.010	0.373/0.000	0.883/0.560	0.799/0.320	0.538/0.000	0.505/0.010	0.519/0.000	0.503/0.000	0.502/0.000	0.516/0.000	0.560/0.078
DwtDctSvd	0.525/0.170	0.526/0.160	0.993/0.960	0.768/0.000	1.000/1.000	1.000/1.000	0.964/0.710	0.845/0.180	0.843/0.170	0.744/0.240	0.517/0.010	0.502/0.020	0.769/0.385
RivaGan	1.000/0.990	1.000/0.990	1.000/0.980	0.479/0.000	1.000/1.000	1.000/1.000	1.000/0.980	0.757/0.080	0.673/0.070	0.830/0.160	0.458/0.000	0.495/0.000	0.808/0.521
SSL	0.999/0.990	1.000/1.000	0.856/0.350	0.997/0.980	0.916/0.630	1.000/1.000	0.679/0.180	0.588/0.070	0.618/0.060	0.637/0.050	0.455/0.000	0.476/0.020	0.768/0.444
ZoDiac	0.991/0.870	0.991/0.840	0.981/0.890	0.732/0.190	0.988/0.720	0.993/0.930	0.984/0.780	0.964/0.370	0.960/0.660	0.956/0.660	0.572/0.060	0.812/0.050	0.910/0.585
Tree-Ring	0.934/0.560	0.930/0.440	0.905/0.350	0.695/0.070	0.917/0.450	0.938/0.570	0.915/0.360	0.873/0.330	0.874/0.250	0.861/0.300	0.546/0.050	0.728/0.050	0.843/0.315
WF (Ours)	1.000/0.990	0.999/0.990	1.000/0.990	0.944/0.630	1.000/0.990	1.000/0.980	0.999/0.980	0.997/0.920	0.996/0.990	0.998/0.940	0.934/0.480	1.000/0.990	**0.989/0.906**

We present Table 5 which is an extension of Table 3. We show the TPR 1%FPR along with the AUC. This new metric gives us a sense about how well our detector is given a specially chosen false positive threshold.

C.2 LOSS WEIGHT ABLATION

Table 6: Perceptual and WDR metric for loss weight ablation. We use the DiffusionDB dataset for this experiment. We highlight the best value for each metric.

Loss Weight	PSNR ↑	SSIM ↑	LPIPS ↓	Pre-Attack ↑	Brightness ↑	Contrast ↑	JPEG ↑	Rotation ↑	G-Noise ↑	G-Blur ↑	BM3D ↑	Bmshj18 ↑	Cheng20 ↑	Zhao 23 ↑	All ↑	All + No Rotation ↑
10^{-2}	25.13	**0.92**	0.121	**0.991**	**0.940**	**0.950**	**0.930**	**0.580**	**0.950**	**0.980**	**0.980**	**0.910**	**0.920**	**0.900**	**0.380**	**0.710**
10^{-3}	25.41	**0.92**	0.121	0.970	0.850	0.870	0.810	0.140	0.830	0.860	0.830	0.760	0.780	0.720	0.090	0.510
10^{-4}	25.58	**0.92**	0.118	0.957	0.810	0.810	0.760	0.110	0.770	0.840	0.750	0.710	0.710	0.620	0.030	0.350
10^{-5}	**25.74**	**0.92**	**0.112**	0.949	0.780	0.770	0.700	0.240	0.680	0.760	0.740	0.650	0.610	0.600	0.140	0.320
10^{-6}	**25.74**	**0.92**	**0.112**	0.940	0.780	0.770	0.720	0.230	0.710	0.770	0.740	0.630	0.610	0.600	0.110	0.320

Table 7: Results for loss weight ablation. We show AUC and TPR@1%FPR across a wide variety of different attacks and perturbations. The dataset used is DiffusionDB. We highlight the best value for each metric.

Method	Post-Attack											
	Brightness	Contrast	JPEG	Rotation	G-Noise	G-Blur	BM3D	Bmshj18	Cheng20	Zhao23	All	All w/o Rot.
10^{-2}	**0.991/0.950**	**0.997/0.950**	**0.993/0.920**	**0.881/0.470**	**0.994/0.920**	**0.997/0.940**	**0.996/0.910**	**0.994/0.910**	**0.985/0.830**	**0.987/0.830**	**0.806/0.220**	**0.947/0.800**
10^{-3}	0.984/0.920	0.989/0.890	0.980/0.860	0.731/0.210	0.982/0.860	0.986/0.860	0.987/0.790	0.977/0.810	0.953/0.730	0.961/0.730	0.636/0.090	0.904/0.590
10^{-4}	0.975/0.890	0.980/0.880	0.976/0.790	0.714/0.170	0.966/0.720	0.979/0.800	0.976/0.740	0.956/0.780	0.938/0.670	0.944/0.640	0.576/0.030	0.831/0.310
10^{-5}	0.951/0.740	0.944/0.680	0.923/0.510	0.687/0.150	0.902/0.520	0.940/0.540	0.916/0.500	0.898/0.570	0.875/0.420	0.874/0.500	0.559/0.05	0.705/0.220
10^{-6}	0.951/0.740	0.944/0.670	0.928/0.500	0.685/0.150	0.890/0.500	0.940/0.560	0.918/0.510	0.898/0.550	0.875/0.400	0.874/0.500	0.564/0.050	0.717/0.160

We present our results in Table 6 and 7. We use the hyperparameters listed in Appendix B. We note that for these experiments we use a SSIM threshold of 0.92.

Our results indicate a general trade off between perceptual quality and detectability/robustness with the loss weight. That is higher loss weights have lower perceptual quality (PSNR, LPIPS) but are better in robustness metrics (for AUC, TPR1%FPR, WDR).

C.3 WATERMARK RADIUS ABLATION

Table 8: Perceptual and WDR metric for watermark radius ablation. We use the DiffusionDB dataset for this experiment. We highlight the best value for each metric.

Watermark Radius	PSNR ↑	SSIM ↑	LPIPS ↓	Pre-Attack ↑	Brightness ↑	Contrast ↑	JPEG ↑	Rotation ↑	G-Noise ↑	G-Blur ↑	BM3D ↑	Bmshj18 ↑	Cheng20 ↑	Zhao23 ↑	All ↑	All + No Rotation ↑
5	**25.69**	**0.92**	0.121	0.958	0.770	0.810	0.740	0.290	0.810	0.850	0.820	0.710	0.760	0.720	0.140	0.480
10	25.13	**0.92**	0.121	0.991	0.940	0.950	0.930	0.580	0.950	0.980	0.980	0.910	0.920	0.900	0.380	0.710
15	25.22	**0.92**	0.117	0.998	**0.990**	0.980	0.990	0.340	0.980	**1.000**	**1.000**	0.970	**0.990**	0.920	0.220	0.810
20	25.06	**0.92**	**0.095**	**0.999**	**0.990**	**0.990**	**1.000**	**0.770**	**0.990**	**0.990**	**1.000**	**0.990**	**0.990**	**0.990**	**0.540**	**0.890**

Table 9: Results for watermark radius ablation. We show AUC and TPR@1%FPR across a wide variety of different attacks and perturbations. The dataset used is DiffusionDB. We highlight the best value for each metric.

Method	Post-Attack											
	Brightness	**Contrast**	**JPEG**	**Rotation**	**G-Noise**	**G-Blur**	**BM3D**	**Bmshj18**	**Cheng20**	**Zhao23**	**All**	**All w/o Rot.**
5	0.977/0.680	0.980/0.770	0.970/0.640	0.798/0.280	0.975/0.670	0.974/0.780	0.980/0.810	0.957/0.690	0.964/0.490	0.949/0.570	0.712/0.220	0.885/0.370
10	0.991/0.950	0.997/0.950	0.993/0.920	0.881/0.470	0.994/0.920	0.997/0.940	0.996/0.910	0.994/0.910	0.985/0.830	0.987/0.830	0.806/0.220	0.947/0.800
15	**1.000/1.000**	**1.000/1.000**	0.999/**0.990**	0.850/**0.500**	**0.998**/0.980	**1.000/1.000**	**1.000/1.000**	**1.000/0.990**	**0.999/0.990**	0.994/0.970	0.806/0.300	**0.969/0.850**
20	0.999/0.990	**1.000**/0.990	1.000/**0.990**	**0.960**/0.360	0.996/**0.990**	**1.000/1.000**	**1.000/1.000**	**0.999/0.990**	**0.999/0.990**	**0.996/0.990**	**0.874/0.450**	0.963/**0.850**

In this ablation we modify the radius of our learned watermark and observe the corresponding results. We present our results in Table 8 and 9. We use the hyperparameters listed in Appendix B. We note that for these experiments we use a SSIM threshold of 0.92.

We observe that increasing the watermark radius leads to a more robust watermark. This makes sense as the watermark assumes more ”area”. However, what is slightly surprising is that the LPIPS seems to become better with a larger watermark radius. So while PSNR suffers, we can understand this as our model creating more realistic images that differ from the original image. A potential reason for this is that a larger watermark means that we have more control over the latent.

C.4 Model Architecture Ablation

Table 10: Perceptual and WDR metric for model architecture ablation. We use the DiffusionDB dataset for this experiment. We highlight the best value for each metric.

Architecture	PSNR ↑	SSIM ↑	LPIPS ↓	Pre-Attack ↑	Brightness ↑	Contrast ↑	JPEG ↑	Rotation ↑	G-Noise ↑	G-Blur ↑	BM3D ↑	Bmshj18 ↑	Cheng20 ↑	Zhao23 ↑	All ↑	All + No Rotation ↑
MLP	25.91	**0.92**	0.111	0.908	0.630	0.600	0.410	0.080	0.520	0.630	0.540	0.460	0.420	0.430	0.000	0.090
Residual Flow	25.13	**0.92**	0.121	**0.991**	**0.940**	**0.950**	**0.930**	**0.580**	**0.950**	**0.980**	**0.980**	**0.910**	**0.920**	**0.900**	**0.380**	**0.710**
UNet	**26.00**	**0.92**	**0.107**	0.501	0	0	0	0	0	0	0	0	0	0	0	0

Table 11: Results for model architecture ablation. We show AUC and TPR@1%FPR across a wide variety of different attacks and perturbations. The dataset used is DiffusionDB. We highlight the best value for each metric.

Method	Post-Attack											
	Brightness	**Contrast**	**JPEG**	**Rotation**	**G-Noise**	**G-Blur**	**BM3D**	**Bmshj18**	**Cheng20**	**Zhao23**	**All**	**All w/o Rot.**
MLP	0.922/0.554	0.910/0.485	0.863/0.386	0.540/0.030	0.880/0.505	0.919/0.495	0.900/0.465	0.841/0.505	0.827/0.396	0.854/0.327	0.485/0.000	0.663/0.069
Residual Flow	**0.991/0.950**	**0.997/0.950**	**0.993/0.920**	**0.881/0.470**	**0.994/0.920**	**0.997/0.940**	**0.996/0.910**	**0.994/0.910**	**0.985/0.830**	**0.987/0.830**	**0.806/0.220**	**0.947/0.800**
UNet	0.553/0.030	0.538/0.050	0.541/0.000	0.499/0.020	0.539/0.020	0.562/0.010	0.518/0.000	0.493/0.000	0.514/0.000	0.550/0.000	0.494/0.010	0.536/0.000

In this ablation we try various generative model architectures for parameterizing our learned watermark. We present our results in Table 10 and 11. We use the hyperparameters listed in Appendix B. We note that for these experiments we use a SSIM threshold of 0.92.

We observe that the Residual Flow architecture yields the best results in terms of robustness although UNet and MLP do slightly better on perceptual metrics. The biggest problem with the UNet architecture is that the learned watermark is simply too weak. That is, the watermarked images are not statistically separable from the non-watermarked images. This can be observed by the 50% AUC and 0 WDR. While MLP is slightly better, it still falls short of the Residual Flow Architecture.

C.5 A Note on Initializing Patch to Tree-Ring

We found in earlier iterations of our work that not initializing with a tree-ring patch produced significantly worse results. Furthermore, adding the tree-ring patch to the learned patch was also worse. This is founded on the hypothesis that the latent with the tree-ring watermark is already a strong starting point and our mapping simply adjusts it as needed to trade off robustness and quality.

C.6 Varying SSIM Threshold

Our results are tabulated in Figure **??**. We present an additional ablation which involves varying the SSIM threshold used for adaptive enhancement. We obviously expect the image quality metrics to get better as we increase the SSIM threshold (note both the PSNR and LPIPS metric). The perhaps more surprising story is the little drop-off in robustness. Between an SSIM of 0.92 and 0.95, there is

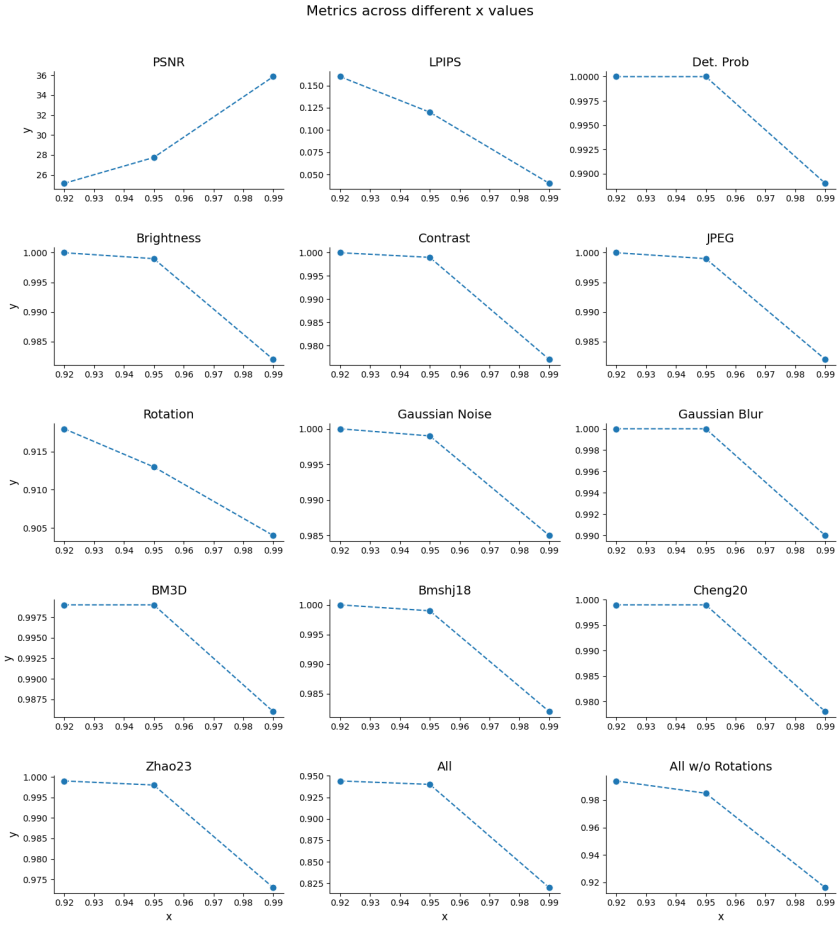


Figure 4: We graph image quality and robustness as a function of various SSIM thresholds used for adaptive enhancement. In this figure, all robustness metrics are in AUC and the thresholds we test are $0.92, 0.95, 0.99$.

little to no drop in robustness quality. The difference starts to become slightly more noticeable when we go up to 0.99 but is still relatively good compare to our baselines. We hypothesize that because adaptive enhancement is similar to an adversarial attack (though beneficial for us in this case), it does not deter WaterFlow which is by nature incredibly robust.

D METRIC DETAILS

Image Quality: We calculate the Peak Signal-to-Noise Ratio (PSNR) between the watermarked image, Structural Similarity Index (SSIM) Wang et al. (2004), and Learned Perceptual Image Patch Similarity (LPIPS) metric Zhang et al. (2018).

Robustness: For measuring the watermark robustness, we report average Watermark Detection Rate (WDR). Given the returned p-value of an image, we consider an image watermarked if the detection probability is greater than some threshold p^*. In our experiments we use $p^* = 0.9$ for WaterFlow, Tree-Ring, and ZoDiac. We change the threshold for the rest of the baselines as detailed in Appendix B.3. We also report the Area under the curve (AUC) along with the TPR@1%FPR or the true positive rate given we want 1% false positive rate (latter metric found in Appendix C.1).

Time Efficiency: We measure the average time needed to watermark a single image.

E LIMITATIONS

Our method relies on a pre-trained stable diffusion generative model. While this model shows strong performance across domains, its applicability to specific image types, like medical images, remains uncertain. Additionally, our method relies on one open-source generative model so it is unclear if it will adapt to closed-source models. Moreover, the assumption of diffusion generative models which have a latent space and are invertible limits the applicability of our method to other SOTA generative models that are autoregressive and lack these features. Thus, while promising, further research is needed to assess its effectiveness across diverse generative models and image domains. We also notice that in some cases our model produces artifacts on the generated image. However, this can be optimized for with the loss weight.

F FUTURE WORK

In our future works, we aim to extend our testing to encompass a broader range of datasets as well as explore more robustness attacks in real-world scenarios to assess the resilience of our method against adversarial challenges. Moreover, while our current methodology assumes that pre-trained generative models maintain the quality achieved through the original diffusion objective, future investigations could explore fine-tuning strategies that integrate the diffusion objective with watermarking objectives.

G SAMPLE IMAGES

In this section we present some example images.

Figure 5: Examples results on DiffusionDB.

Figure 6: Examples results on DiffusionDB showing the image before it is watermarked, the output after watermarking, and finally the adaptive enhancement image.

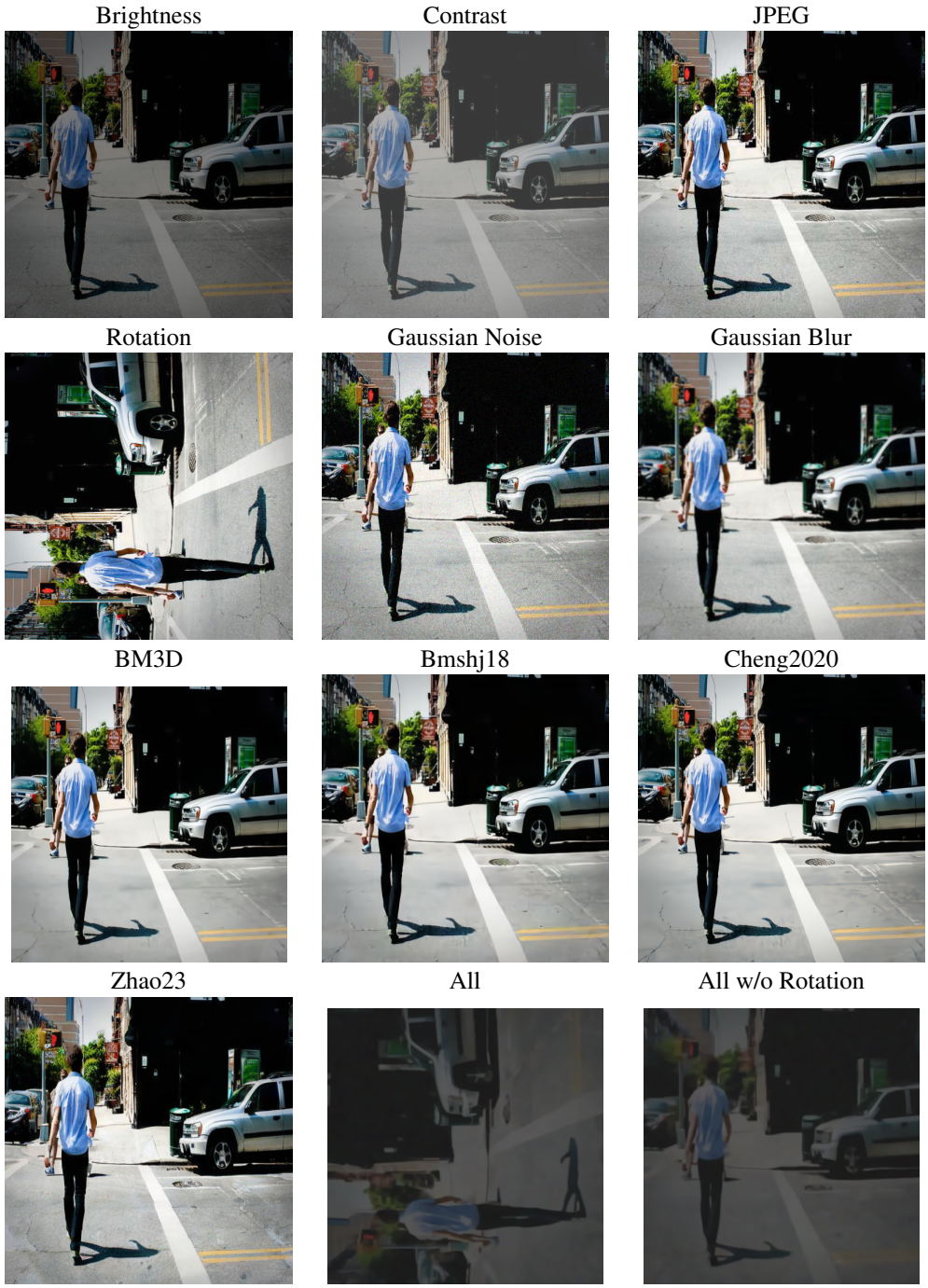


Figure 7: Examples of attacks on watermarked iamge from MS-COCO dataset.