Internship Report

Ismail Labiad

October 21, 2024

Adversarial Attacks and Detection of Fake Images

Contents

1 Introduction

AI-generated images, while beneficial in various contexts, have also been leveraged for malicious purposes. Deepfakes / AI-generated videos or images that convincingly alter a subject's appearance or behavior can be used in political propaganda, identity theft, and even cybercrimes [1, 2]. Moreover, the recent EU AI act states that AI audio-visual and text outputs should have machine readable labels to distinguish them from human outputs and deepfakes [3, 4]. Therefore, it is necessary to be able to distinguish between real and fake images.

Different approaches have been developed to trace AI generated content. The simplest used method involves modifying the **metadata** of an image to store information about it origin, while this approach is robust to any image transformation, it can be easily removed or manipulated (e.g. with a screenshot). The straightforward deep learning approach is **passive detection** where the problem is formulated as a binary classification and a model is trained to separate between real and fake images [5], but the question of transfer capabilities to unseen generators and the robustness of these detectors to out-of-distribution data (through resizing and compression) still remains unanswered. With the increase of computational power, methods such as database **indexing** and retrieval can be used, where we store a compressed version of each generated image and look up a proxy distance at test time to determine if we have generated the given image [6]. Another approach involves **watermarking** the generated images, or the generation process itself, by embedding a secret message in the image and identifying the extracted message at test time [7]. There exists other method such as **on device** techniques which generates an irreversible hash for a real captured image that would change with the slightest modification of the original image, making it possible to tell if an image has been taken by a given device and has not been modified [8].

But with the development of these detection methods, new adversarial attacks [9] have been designed to fool them. It has been shown that deep classifiers can be easily attacked to miss-classify an image by introducing a small perturbation to the input image [10]. These attacks can either have access to the classification model and its gradients known as **white box** attacks such as FGSM, PGD, DeepFool [11], etc. But, in practice most of the detectors are not accessible to the public, and we might only have access to the fake probability or the classification result in which case gradient free attacks or **black-box** attacks can be used to generate adversarial examples either by limiting the search space to generate a specific form of noise or by estimating the gradients from the outputs [10], [12], [13]. The last family of **no-box** attacks cover a broad range of transformations which only takes images as input and aims to change the distribution of images to the one seen during training, which can be done through compression (JPEG or neural autoencoders), blur, transferring the attack from a locally trained classifier etc. But the most effective use case is against watermarking extractors, where diffusion models are used to remove the embedded watermark while not significantly modifying the image [14].

In this work, we focus on mitigating the effectiveness of black-box adversarial attacks on passive detectors such as Universal Fake Detector [15] and no-box diffusion attacks on watermarking approaches such Stable Signature [16]. We show that we can train a classifier to detect known black-box, such as square attack, and no-box attacks, such as DiffPure [17], if we have access to a few thousand attacked samples. We also show that training a detector on one attack method doesn't transfer to other unseen attacks that generates a different noise, as the new Log-normal optimization algorithm [18]. We introduce in section 4 a novel approach that uses text watermarking techniques to watermark images through the quantized latent space of VQGAN [19] and show

that it presents a noticeable robustness toward non-geometrical image transformations. Finally, in section 5 we perform a comprehensive evaluation of possible ways to integrate these detectors of attacks and combine all detectors into one ultimate detector and show that it is more robust to adversarial attacks.

As a summary, our contributions are:

- We show that the newly introduced gradient free optimizer Log-normal can be used as black-box attack on fake detectors with great success rate up to 99%.
- We show that we can detect no-box attacks for known purifiers.
- We show that we can detect square attack from a few samples, but not transfer to detecting log-normal attacked images.
- We introduce a novel watermarking approach that uses text watermarking and test it robustness against various non-geometrical transformations.
- We show that combining various detectors achieves a better accuracy overall on detecting different fake attacked images, and that it is more robust than the individual passive detector.

2 Related work

2.1 Passive fake detectors

Traditionally, researchers have proposed to detected fake generated images using handcrafted features like forensic features such as compression artifacts[20], or features extracted using image transformations such as SVD applied to the discrete cosine transform of the image [21].

Recent studies leverage deep learning to learn to extract useful features and generation artifacts directly from the data. The landmark study by [5] proves that we can detect GAN generated images which are CNN based and transfer to unseen generators during training, by simply employing Blur and JPEG data augmentations during training. The authors of [15] propose UnivFD that uses the feature space of the frozen pretrained large model on images CLIP [22] and show that we can distinguish between AI generated images and real images by simply adding a final linear classification layer. The work done in [23] show that a teacher student architecture with a learnable adversarial augmentation model allow to obtain more transfer to various unseen images generated by GANs and also by recent diffusion models.

More recent works rely on textures analysis by using frozen SRM filters [24] as additional input features, such as the work in [25] that integrates inter-pixel correlation contrast between rich and poor texture regions within an image to boost performance and [26] which leverages multiple experts to simultaneously extract visual artifacts and noise patterns.

2.2 Watermarking techniques

Image watermarking [27] consists of inserting a mark/message into an image using a secret key. The information should then be retrieved even if the image has undergone transformations such as compression, contrast change, crops, rotations, etc. We usually distinguish zero-bit watermarking, where we encode the presence or absence of the mark, from multi-bit watermarking where the goal is to encode a bitstring message.

Traditional image watermarking methods are usually classified in 2 categories, depending on the space on which the watermark is embedded. In Spatial domain techniques, the watermark is encoded by directly modifying pixels (such as simply flipping low-order bits of selected pixels) while frequency domain (or transform domain) techniques alter some frequency coefficients obtained by transforming the image into the frequency domain.

To cite a few pioneering works in spatial domain watermarking, this is the case of [28] where the algorithm utilizes the zero or the minimum points of the histogram of an image and slightly modifies the pixel grayscale values to embed data into the image (this algorithm is also reversible, meaning that one can also retrieve the original, undistorted image). [29] casts the watermark by slightly modifying the intensity of randomly selected image pixels, which can be done in such a way that the watermark is resistant to JPEG compression and low pass filtering. The detection is carried out by comparing the mean intensity value of the marked pixels against that of the pixels not marked using statistical hypothesis testing.

The second category (and the one that usually provides better robustness) is frequency domain watermarking. The first step is to use a transformation method such as discrete cosine transform (DCT) to compute coefficients that act as descriptor of the original image. The watermark is then added to these coefficients and mapped back onto the original pixel space to generate the watermarked image. The motivation behind it is that these coefficients provide descriptors invariant to the transformations. Among the transform domains that were used in previous watermarking methods, we can cite Discrete Fourier Transform [30], Quaternion Discrete Fourier Transform (QDFT) [31], Discrete Cosine Transform (DCT) [32], SVD-DCT [33], Discrete Wavelet Transform (DWT) [34], DWT-DCT [35].

Deep-learning-based watermarking has emerged as a viable alternative to traditional methods. The networks are often built as encoder/decoder architectures [36], where an encoder embeds a message in the image and a decoder tries to extract it. For instance, HiDDeN [7] jointly trains encoder and decoder networks with noise layers that simulate image perturbations. It uses an adversarial discriminator to improve visual quality, and was extended to arbitrary image resolutions and message lengths by Lee et al. [37]. Distortion Agnostic [38] adds adversarial training that brings robustness to unknown transformations and [39] embed the mark with an attention filter further improving imperceptibility. Finally, ROMark [40] uses robust optimization with worst-case attack as if an adversary were trying to remove the mark. For more details, we refer to the review [41]. This type of methods has its drawbacks: e.g., it is hard to control the embedding distortion, and it is made for decoding and does not translate so well to detection.

2.3 Adversarial attacks

Deep Learning is currently used to perform multiple tasks, such as object recognition, face recognition, and natural language processing. However, Deep Neural Networks (DNNs) are vulnerable to perturbations that alter the network prediction through adversarial examples [9].

Adversarial examples can be generated through various levels of access to the Deep learning models, in the literature we state **white-box** access where we assume that we have access to the model parameters, architecture and/or training data. In **black-box** access, we only have access to the model output without gradients for a specific set of samples or any arbitrary samples. We also note **no-box** access where we assume that we have no access to the model or its training data.

The first class of black-box adversarial attacks rely on estimating the gradients, which inherently

requires more queries. To cite a few works, [42] proposed Zeroth Order Optimization(ZOO) to estimate the gradients of target model in order to produce an adversarial image, [43] propose a two-sided gradient estimation using finite differences, [44] where the authors propose AutoZOOM an adaptive random gradient estimation method for stabilizing query counts and distortion.

The second class focuses on search algorithms, which are able to bypass defenses based on Gradient Masking and Obfuscation, by searching through specific noise patterns to reduce the search space [10], based on genetic approaches [18, 45], or by using the low frequency subspace of the image through the use of the discrete cosine transform (DCT) [12].

No-box on the other hand rely either on transferability through attacking a locally trained model and transferring the attack to the target model, such as shown in [46], or on removing the secret message (watermark) embedded in an input image by producing a visually similar purified output image. The simplest approaches are done through compressing the image using neural autoencoders [47, 48, 19] as tested in [16]. Recent approaches employ diffusion models [49, 50] to purify images by adding noise and denoising to effectively remove the watermark [14, 51].

2.4 Detecting and defending against adversarial attacks

Various techniques have been developed to defend against adversarial attacks [52]. The first category relies on gradient masking, where methods such as distillation [53] aims to make it more difficult to estimate the gradients.

The second category involves training the model on transformed input images and relies on the properties of the transformations used to remove/weaken the attacks. To cite a few works, [54] show that total variation [55] minimization is effective against attacks, [56] first introduced denoising models based on autoencoders, [17, 57] uses GAN or recent diffusion models as denoisers and trains models on denoised/purified images.

The third category consists of simply adding adversarial images in the training data to make the trained model more robust. Authors of [58] propose to augment the training dataset of the target model with adversarial examples transferred from other static pre-trained models, [59] show that adding attacked images by PGD in the training set allow the trained model to be robust against all first-order based adversarial attacks.

The last category try to mitigate adversarial attacks by detecting them. Methods such as [60] introduces a detector based on the distance of the inputted image to it reconstructed version using an autoencoder, a similar approach is done in [61] where the authors propose to train a classifier on filtered images and then flag input images as attacked if the detection result differs between the filtered and non-filtered images. Other works, detect adversarial attacks inside handcrafted feature spaces such as Fourrier [62] or smoothed versions of the image [63].

3 Adversarial attacks and their detection

Our attacks (Section 3.1) consider the fake detector as a black-box, so we do not have gradients. Then, in Section 3.2, we create defense mechanisms by detecting various black-box attacks, showing that the log-normal attack is not detected by a detector created for a classical attack (such as Square Attack) and, therefore, needs new ad hoc detectors.

3.1 Black-box attacking fake detectors

We consider stress tests for fake detectors. More precisely, we add imperceptible noise e to the image x, with this noise chosen so that fake detectors fail: typically x is a fake image detected as fake by a detector $\mathcal{D}$ and we find a small e such that $x + e$ is classified as non-fake by $\mathcal{D}$. Black-box algorithms can be straightforwardly applied to attacking a fake detector $\mathcal{D}$ which returns the estimated probability $\mathcal{D}(x)$ that an image x is fake. We define $loss(e) = \mathcal{D}(e + x)$ and minimize $loss$ using a black-box algorithm on a domain D, typically $[-0.03, 0.03]^t$ where t is the shape of the image tensor[1].

The fake detector we consider is the universal fake detector [15]. Our dataset of fake images is sampled from the test set provided by the authors. IN500- refers to the 500 first classes of ImageNet, IN500+ to the other classes of ImageNet, G refers to Glide, PG refers to ProGAN, BG refers to BigGAN, LDM-200 is defined in the test set of [15].

Critical cases. We consider the detection of attacks (no-box or black-box attacks) aimed at evading fake detectors. So, we are particularly interested in correctly detecting attacks in two cases (1) fake images attacked for making them appear genuine and (2) real, unaltered images (clean).

The other cases are less critical: (1) the case of real images attacked so that they will look fake: this will usually not prevent the original images from being classified as real.

(2) the case of clean fake images : erroneously viewing them as attacked will not prevent a fake detector from working.

Therefore, besides the accuracy on our datasets, we will present the accuracy on the critical part of these datasets.

We first attack the fake detector with the classical Square attack black-box. We give a brief overview here and full details are in appendix A. Nevergrad [64] provides modifiers (noise transformation) dedicated to domains shaped as images. These modifiers can be applied to any black-box optimization algorithm. For example, the prefix *Smooth* means that the algorithm periodically tries to smooth its attack x, if the loss of $Smooth(x)$ is better than the loss of x, then x is replaced by $Smooth(x)$ in the optimization algorithm. We add two additional modifiers specific from adversarial attacks. Both are inspired from SA, though our modifications have, by definition, less horizontal and vertical artifacts. First, G (Great) means that we replace $loss(x)$ by $loss(0.03 \times \text{sign}(x))$ when the allowed norm of the attack is 0.03 for the l^∞ norm. Second, SM (smooth) means that we replace the loss function f by $f(convolve(x))$, where $convolve$ applies a normal blurring with standard deviation 3/8. GSM means that we apply both G and SM. For example, we get GSM-SuperSmoothLognormalDiscreteOnePlusOne by applying G, SM and SuperSmooth as modifiers on top of the standard log-normal method, and various methods as in Tab 2. Results are presented

[1] In this case, we assume that pixels are in the range $[0, 1]$. Due to maximum values in image representations, we might clamp the values of $e + x$.

Table 1: Datasets used in the study. Dataset3 will be used as a test set for the detectors trained on Dataset2 and Dataset4, will be used for testing the transfer of detectors trained on SA attacks to log-normal attacks.

Name	Description
Dataset1	2k images from PG, 2k images from BG and 1k images from LDM-200 for a total of 5k fake images. Used to benchmark variations of log-normal attacks and square attack.
Dataset2-DP	Same as Dataset1, plus 5k real images from IN500- and their purified versions by DiffPure with parameter 0.1
Dataset2-IR	Same as Dataset1, plus 5k real images from IN500- and their purified versions by ImageRephrase with parameter 0.1
Dataset2-SA	Same as Dataset1, plus 5k real images from IN500- and their attacks by SA with budget=10k and $l^\infty = 0.01$. Total = 20k images.
Dataset2-LN	Same as Dataset1, plus 5k real images from IN500- and their attacks by log-normal (algo1) with budget=10k and $l^\infty = 0.01$. Total = 20k images.
Dataset3	1k real images from IN500+ + 500 fake images from PG of different classes than Dataset2 + 500 fake images from P and their attacked counterpart. Total = 4k images.
Dataset3-IR0.1, IR0.2, IR0.3	Purified versions of Dataset3 based on ImageRephrase, with parameter 0.1, 0.2, 0.3.
Dataset3-DP0.1, DP0.2, DP0.3	Purified versions of Dataset3 based on DiffPure, with parameter 0.1, 0.2, 0.3.
Dataset3-SA (L,B)	Real images from Dataset3 + fake images from Dataset3 attacked by SA with various amplitudes ($L \in \{0.01, 0.03, 0.05\}$) and budget ($B \in \{1000, 10000\}$).
Dataset3-LN (L,B)	Real images from Dataset3 + fake images from Dataset3 attacked by LN (algo1 to algo5) with various $L \in \{0.01, 0.03, 0.05\}$ and $B \in \{1000, 10000\}$.
Dtaset4	1k fake images from Dataset3, attacked with log-normal attacks (algo1).

in Tab. 3. Examples of attacked images are presented in Figure 1. Basically, when we have a black-box access to a fake detector, we can attack it either by SA or by log-normal with 10k queries and $l^\infty = 0.03$.

Original algo 1 algo 5 SA

Figure 1: Example of attacked images. All attacks are done with a budget of 10k queries and $l^\infty = 0.03$.

Table 2: Variations of the log-normal algorithm considered.

Algorithms	Alias	Algorithms	Alias
GSM-SuperSmoothLognormalDiscreteOnePlusOne	algo1	LognormalDiscreteOnePlusOne	algo4
G-SuperSmoothLognormalDiscreteOnePlusOne	algo2	GSM-BigLognormalDiscreteOnePlusOne	algo5
SuperSmoothLognormalDiscreteOnePlusOne	algo3	G-BigLognormalDiscreteOnePlusOne	algo6

Table 3: Success rate when attacking the fake detector. The success rate is computed solely on attacks of correctly classified clean images. All attacks have 10k budget and $l^\infty = 0.03$. Inspired by SA, with maximum values of modifications ("G") covering an area ("SM"), Algo5 performs best among LogNormal variants. Compared to SA, it is less specific from images and it has a self-adaptive mutation rate starting at a high value ("Big")

Algorithm	SA	algo1	algo2	algo3	algo4	algo5	algo6
Success rate on dataset1	100%	91.0%	94.1%	64.6%	82.0%	99.2%	99.1%

3.2 From attack to defense: detecting attacks

3.2.1 No-box attacks (purifiers) and their detection

While watermarking techniques can be made robust against some image alterations such as resizing or JPEG compression as shown by [16], they remain vulnerable to no-box purification attacks [14] that destroy the hidden message. Additionally, the distribution change introduced by no-box purification can impact negatively the performance of a fake detector, as shown in Table 4. We will next show that one can easily detect these attacks if we have access to the purification method.

We consider two purifiers used by [65]: DiffPure based on guided diffusion and ImageRephrase based on latent diffusion. We consider the strength/steps parameter in the set {0.1, 0.2, 0.3}. These two purifiers lead to an average PSNR (Peak Signal-to-Noise Ratio) between 20 and 30, which is a noticeable yet acceptable quality loss. As expected, the PSNR of the distortion increases with the strength parameter.

Table 4: The second row shows the false negative rate (FNR) at a threshold of 0.5 for the universal fake detector on clean Dataset1 consisting of only fake images and different purified versions of this dataset. DP stands for DiffPure while IR stands for ImageRephrase. The third row corresponds to the average PSNR.

	Attacks						
	Clean	DP 0.1	DP 0.2	DP 0.3	IR 0.1	IR 0.2	IR 0.3
FNR	5.2%	23.42%	34.4%	40.1%	11.3%	16.8%	26.2%
PSNR	NA	27.9	24.4	22.1	26.6	24.4	22.7

For detecting the no-box attacks, we train a classifier, specified in table 14, on Dataset2-DP or Dataset2-IR. We experiment with both ResNet50 and SRnet [66] and the latter performs better overall. We use Dataset3 as a test set (so that the distributions of images differ) with images attacked by same purifier but with different parameters in $\{0.1, 0.2, 0.3\}$. The results of the detection of latent purifiers and guided diffusion purifiers are shown in Table 5 and Table 6. We achieve less than 5% FPR and FNR across the hold-out training dataset and the critical part of testset, although the FPR is relatively higher for the full testset that can be explained by having a sort of transfer from detecting DP or IR to detecting images generated by G.

In short, training on a purifier with a specific parameter allows us to detect that same purifier with different parameters and for different image distributions.

3.2.2 The detection of black-box attacks

For detecting square attacks, we train a deep net using a similar setup, as the SRnet model still performs better than ResNet in this scenario. We also employ data augmentation techniques: horizontal flip, random crop and color jitter which empirically makes the model more robust to different attack parameters, while allowing us to train using only one attack parameter budget=10k and $l^\infty = 0.01$. The training is done on Dataset2-SA. For testing, we use Dataset3-SA using different parameters than those used during training. We then test the transfer of the SA detector on Dataset4, corresponding to Log-normal attacks. Table 7 summarizes the results of the SA detector, for detecting SA and as a detector of log-normal attacks. We observe that the transfer to detecting

log-normal attacks is very poor. So, we need to include such attacks in our training for improving the defense.

A new detector for log-normal attacks. We have seen that detectors of SA do not detect our log-normal attacks. For detecting log-normal attacks, we use the same setup as in section 3.2.2 for creating a new detector. Dataset2-LN is used during training with the attacked images now obtained with GSM-SuperSmoothLognormalDiscreteOnePlusOne budget=10k and $l^\infty = 0.01$ and again, the dataset is split to 80% train, 10% test, 10% validation. For testing, we use Dataset3 with the images attacked using different variations of log-normal and parameters (Budget and l^∞). Table 8 presents the results of the log-normal detector: we observe that the learning was made on images attacked by algo1 only and we get positive results for all log-normal variants.

Table 5: False positive and false negative rates for purified images detection at a threshold of (0.5) alongside with the AUC score. The purified training images correspond to the latent-purifier (class "ImageRephrase") with parameter 0.1. We observe good results, in particular for critical cases.

	Dataset	FPR↓	FNR↓	AUC↑
(same dist)	Dataset2(param=IR0.1) hold-out	0.9%	0.3%	0.99
	Dataset 3, full (critical and non critical images)			
	Dataset3 (param=IR0.1)	23.1%	0.1%	0.96
(different dist)	Dataset3 (param=IR0.2)	23.1%	0.2%	0.96
	Dataset3 (param=IR0.3)	23.1%	0.2%	0.95
	Dataset 3, critical part			
	Dataset3 (param=IR0.1)	1.7%	0.0%	0.99
(different dist)	Dataset3 (param=IR0.2)	1.7%	0.0%	0.99
	Dataset3 (param=IR0.3)	1.7%	0.0%	0.99

Table 6: False positive and false negative rates for purified images detection at a threshold of (0.5) alongside with the AUC score. The purified training images correspond to the guided-diffusion purifier (class "DiffPure") with parameter 0.1. We observe good results, in particular when we restrict the analysis to critical cases (see the specification of critical cases in section 3.1).

	Dataset	FPR↓	FNR↓	AUC↑
(same dist)	Dataset2(param=DP0.1) hold-out	1.9%	3.9%	0.99
	Dataset 3, full (critical and non-critical)			
	Dataset3 (param=DP0.1)	23.0%	3.7%	0.88
(different dist)	Dataset3 (param=DP0.2)	23.0%	4.0%	0.87
	Dataset3 (param=DP0.3)	23.0%	4.2%	0.87
	Dataset 3, critical part			
	Dataset3 (param=DP0.1)	2.2%	3.0%	0.99
(different dist)	Dataset3 (param=DP0.2)	2.2%	4.6%	0.99
	Dataset3 (param=DP0.3)	2.2%	5.4%	0.99

Table 7: False positive and false negative rates for square attack detection at a threshold of (0.5) alongside with the AUC score. B stands for the budget used for the attack and L stands for the accepted l^∞ distance. Last row: (clearly failed) transfer to log-normal detection.

Dataset		FPR↓	FNR↓	AUC↑
(same dist)	Dataset2-SA hold-out	1.8%	0.5%	0.99
(different dist)	Dataset3-SA (B=10k, L=0.01)	0.8%	0.9%	0.99
	Dataset3-SA (B=10k, L=0.03)	0.8%	0.0%	0.99
	Dataset3-SA (B=10k, L=0.05)	0.8%	0.0%	0.99
	Dataset3-SA (B=1k, L=0.01)	0.8%	0.0%	0.99
	Dataset3-SA (B=1k, L=0.03)	0.8%	0.0%	0.99
	Dataset3-SA (B=1k, L=0.05)	0.8%	0.0%	0.99
(transfer to log-normal)	Dataset4	NA	79.5%	NA

Table 8: False positive and false negative rates for log-normal detection at a threshold of (0.5) alongside with the AUC score. Algorithms are reported in Table 2, B stands for the budget used for the attack and L stands for the accepted l^∞ distance. LN stands for log-normal.

Dataset		FPR↓	FNR↓	AUC↑
(same dist)	Dataset2-LN hold-out	0.7%	2.1%	0.99
(different dist)	Dataset3-LN (algo1, B=10k, L=0.01)	4.1%	5.4%	0.98
	Dataset3-LN (algo1, B=10k, L=0.03)	4.1%	2.3%	0.99
	Dataset3-LN (algo1, B=10k, L=0.05)	4.1%	2.9%	0.99
	Dataset3-LN (algo2, B=10k, L=0.03)	4.1%	4.1%	0.98
	Dataset3-LN (algo3, B=10k, L=0.03)	4.1%	11.7%	0.95
	Dataset3-LN (algo4, B=10k, L=0.03)	4.1%	1.6%	0.99
	Dataset3-LN (algo5, B=10k, L=0.03)	4.1%	0.4%	0.99

3.3 Classifier limitation

Formulating the detection of AI generated images, or adversarial attacks as a binary classification problem has it own limitations. First, separating the distributions of "real" and "fake" is a poorly defined problem as we don't even know what exactly is the "real" distribution and recent AI models are able to generated photo realistic images that are hard to distinguish even for humans [26] making passive detection probably an unachievable goal in the long run. Second, most fake detectors rely on detecting 'artifacts' either introduced by upsampling layers in GANs and some diffusion models [67], or introduced by the 'camera' in real images [25]. This means that one can easily craft adversarial examples for these models, either to bypass them or for spoofing attacks.

Moreover, as the distributions of "real" and "fake" are hard to distinguish, one should be very careful when training a Deep Network for the binary classification task as any unintended artifacts introduced in the data processing will be considered first and the final trained detector will either be biased or entirely flawed. For example, DIRE [68] which uses a ResNet50 as a classifier predicts the input image as fake if it is JPEG compressed and as real if the image is in PNG format, due to real images being in JPEG format in their training dataset while fake images are not. Resizing

and compression artifacts are present in a lot of detectors, and they should be taken into account during the training, as shown in [69].

4 Text watermarking for Images

In this section, we explore the use of the recent text watermarking techniques [70], [71] on images.

The majority of image watermarking techniques [72, 7] relies on introducing an unnoticeable noise to the image and retrieving the secret message (or watermark) in the pixel space. And in all these methods, the watermark extractor is trained with image augmentations to increase it robustness toward these augmentations seen during training [16].

The work done in Tree-Ring [73] explores watermarking the latent space of Latent Diffusion Models that results in a 'strong' modification in the pixel space which inherently give the approach robustness toward a wide range of image augmentations at the cost of the watermarked image being very different from the non-watermarked version.

Our method takes a similar approach to Tree-Ring by watermarking the codes of the quantized latent representation of an image via VQGAN [19]. We show in the next subsections that by doing this we don't suffer from a "strong" change to the outputted image, and that our approach allow us to watermark already existing images in addition to doing in-generation watermarking.

4.1 Method

4.1.1 Method details

The main idea is to obtain the codes $z_c \in E^{H' \times W'}$ of the quantized latent representation $z_q \in \mathbb{R}^{H' \times W' \times C}$ of an input images I, where E is an integer interval for example $[1 \dots 1024]$ and (H', W') are 16, 8, 4 smaller than the original image dimensions (H, W) depending on the compression of the encoder. After obtaining the codes z_c we apply the watermark by changing the value of some codes to a different close value in terms of Euclidean distance of their corresponding latent vector to obtain $z_{c,w}$. Different example approaches to change the codes are explored in the next subsection 4.1.2.

We then decode the watermarked latent representation $z_{q,w}$ to obtain the watermarked image I_w in the pixel space.

To retrieve the watermark in an image, we first retrieve the corresponding codes z'_c and compute a statistical test based on the bias we introduced to $z_{c,w}$ then make a decision based on the p-value. Figure 2 shows a diagram of our method.

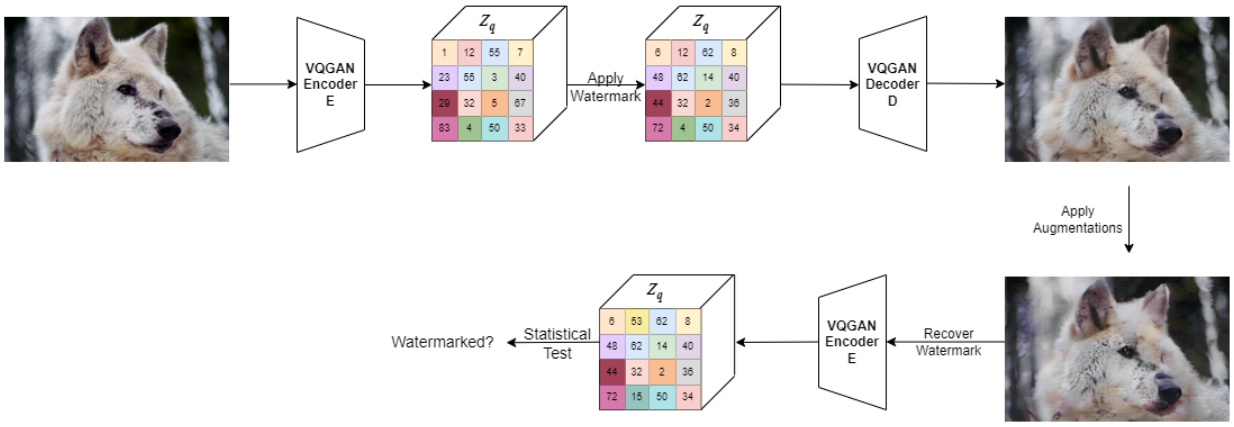


Figure 2: Diagram of our approach.

4.1.2 Choice of watermark

Hard green list We start by picking a set of codes, that we call **green list**. Then for each code of the latent representation z_c we change it to the nearest code in the green list according to the l_2 distance of their corresponding vectors to obtain the watermarked latent representation $z_{c,w}$. We then decode the $z_{q,w}$ obtained from $z_{c,w}$ to obtain the watermarked image.

Detecting the watermark: We attribute to each code a variable $X \in \{0,1\}$ that represents that the code is in the green list ($X = 1$) or not ($X = 0$) and we assume that these variables are **independent** Bernoulli random variables with parameter $p =$ probability of being in the greenlist which we approximate by $p = \frac{\text{green list length}}{codebooklength}$. We then count the number of codes **N** in the green list recovered in z'_c, which under our assumption follows a Binomial distribution with parameters ($H' \times W'$, p) and compute the probability of getting a high **N** at random which correspond to the acceptable false positive rate.

$$\text{FPR} = \mathbb{P}(N > \tau) = \sum_{i=\tau}^{H' \times W'} \binom{H' \times W'}{i} p^i (1-p)^{H' \times W' - i}$$

For example, we can pick the green list to be the even codes in our codebook. Figure 3 shows an example of original, reconstructed without watermark and watermarked image along with their p-value. Their corresponding latent codes can be found in Figure 13 in the Appendix.

Original
p-value=0.36

Reconstructed
p-value=0.36

Watermarked
p-value=1.7e-175

Figure 3: Example of original, reconstructed without watermark and watermarked image, with their corresponding p-value. The model used is VQGQN_f16_1024 with our finetuning.

Soft green list We start by attributing an artificial probability $q^k_{i,j}$ to each couple $(c_{i,j}, c_k)$ as follows:

$$q^k_{i,j} = \exp\left(-\|u_{i,j} - v_k\|^2\right)$$

Where $c_{i,j}$ is the code at position (i,j) in the latent representation z_c and $u_{i,j}$ is it corresponding vector. And where c_k iterates over the codes in the codebook and v_k is it corresponding vector.

We then rely on text watermarking techniques as in [70], [71] to watermark the generation of $z_{c,w}$ given the constructed probabilities q. We won't go into details of these techniques, but we present below a simplified version of text watermarking that we will use in our experiments.

For example, we can use a soft green list approach where we pick the even codes as our green list (or generate a random one from a secret code s as a seed). We then add a factor δ to each value $q^k_{i,j}$, if c_k is in the green list, when picking the highest probability candidate code for the position (i, j) (which is $c_{i,j}$ if $delta = 0$). Therefore, biasing the generation of $z_{c,w}$ but making sure that "important" codes that have no other vector close to them don't get replaced, this should help avoid hurting the image quality.

Figure 4 shows an example of original, watermarked image and p-value corresponding to each one. Their corresponding latent codes can be found in Figure 14 in the Appendix.

More watermarked images can be found in Figure 15.

Original
p-value=0.64

Reconstructed
p-value=0.64

Watermarked
p-value=4.1e-62

Figure 4: Example of original, reconstructed without watermark and watermarked image, with their corresponding p-value. The model used is VQGQN_f16_1024 with our finetuning. The green list is generated with 2024 as seed.

4.1.3 Idempotence

In order for our method to work, we need to be able to retrieve the same codes we modified. Formally, we want that the function $z \rightarrow$ Encoder(Decoder(z)) to be close to the identity on the vectors of our codebook.

We finetune the base VQGAN to take into account this constraint through a surrogate loss term, as we can't back propagate from comparing the error rate on the integers codes. The final loss is a sum of all the terms :

$$\text{new\_loss} = \text{loss}_{\text{VQGAN}} + \lambda_1 MSE(sg[z], z_{rec}) + \lambda_2 MSE(z, sg[z_{rec}])$$

Where z correspond to the output of the encoder just before the quantization and z_{rec} to the output of the Encoder(Decoder(z)) just before the quantization. In practice, we set $\lambda_1 = 100$ and $\lambda_2 = 1$.

In practice, we finetune the base model for 2 epochs on 10k images from ImageNet first 500 classes, and we resize the images and keep the aspect ratio then crop to the target size of 256×256. During testing, we only crop the image to the nearest lower multiple of the compression factor. For example, if the input image is of size (H, W) = (333, 500) it is cropped to the size (320, 496) for a compression factor $f = 16$.
Figure 5 shows a comparison of the average PSNR and error rate of the retrieved codes of the base and finetuned models on 100 random images from ImageNet last 500 classes. It can be seen that we largely improve the recovery of the codes while not hurting the quality of the outputted images.

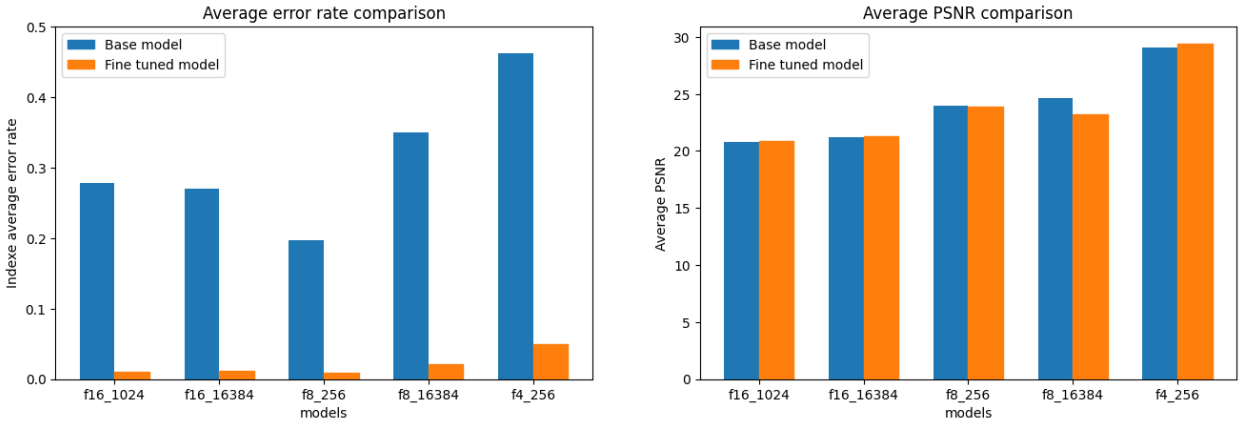


Figure 5: Comparison of the average PSNR and error rate between the base and finetuned versions of VQGAN.

Training for robustness Embedding the watermark in the latent space introduces a semantic change to the watermarked image in the pixel space which should be inherently robust to some non-geometrical transformations such as JPEG compression, resizing, blur, etc. But we can also increase this robustness by adding augmentations during training, where we finetune the base-model so that $z \rightarrow$ Encoder(Augmentation(Decoder(z))) is close to the identity on the vectors of the codebook.

In our implementation we work with the original image size after cropping it to the nearest multiple of 16 (H_{16}, W_{16}) (in case of a model with f16 compression factor), instead of resizing the image to a fixed image size target (for example (256, 256)). Although both approaches are viable, the first one allows us to have a bigger (H', W') which improves our statistical test and, more importantly, allow us to be robust to Cropping as we won't stretch the image. But contrary to the second approach, this doesn't allow us to compare the latent codes of an image and it resized version, so in our experiments we resize the image back to (H_{16}, W_{16}). In practice, we won't have access to the original size of the image, a solution can be to run a search on possible values of (H, W) and flag an image as watermarked if we find a p-value below a set threshold.

In generation watermarking The method detailed above works for any already existing input image, but we can also watermark images during the generation process either by applying the watermark after the generation of the latent representation z_c with a transformer, or by limiting the possible generated codes by the transformer in the case of the hard green list approach or by finetuning the transformer on the biased watermarked codes in the case of soft green list approach.

4.2 Experiments

In this section, we present robustness and image quality result of our approach.

4.2.1 Datasets, training details and metrics

To simplify, the figures will only use and report results on VQGAN_f16_1024 and VQGAN_f16_16384.

We finetune 2 versions of each base model, the first VQGAN_f16_1024_ft without augmentations and a second version VQGAN_f16_1024_ft_aug with a few augmentations during training (JPEG + resizing) and test on seen and unseen augmentations during testing.

Dataset We use 10k randomly selected images from the first 500 classes of ImageNet as are training dataset, where we resize and randomly crop to the target size (256, 256). For all our testing we use 100 randomly selected images from the last 500 classes of ImageNet.

Training hyperparameters The finetuning is done using the Adam optimizer same as in the original paper [19] with a base learning rate of 4.5e-6 and a batch size of 4. The finetuning is run for a total of 2 epochs, and it lasts 30 min on a single GPU. For the VQGAN_f16_1024_ft_aug training, we apply random resizing with probability 0.2 and a resize strength randomly selected in $[1.0 \dots 0.2]$, and apply random JPEG compression with a probability of 0.2 and a compression value randomly chosen in $[100 \dots 10]$.

Metrics We report average code recovery error rate as our signal quality in addition to the true positive rate at 1% false positive rate (TPR@1FPR) by setting the p-value threshold to 1e-2. For image quality we report PSNR, LPIPS, SSIM and SIFID between the original input image and the outputted watermarked image. And as we are limited by the quality output of the autoencoder, we will also report the image quality loss between the reconstructed image without the watermark and the watermarked version.

We consider the following image transformations to test the robustness of our approach: resizing (without stretching), JPEG, Gaussian blur, pixelate, brightness, crop.

In all the experiments we use the soft green list approach introduced in section 4.1.2 as our watermarking method, we pick half the codes from the codebook to be in our fixed green list (by setting the seed=2024) and estimate p=0.5 the probability of being in the green list.

4.3 Results

4.3.1 Error rate (bit accuracy)

Resizing We resize each image dimension by a factor in $[1.0 \dots 0.2]$ while keeping the aspect ratio, and we then resize the image back to it original size. Both resize operation use bilinear interpolation.

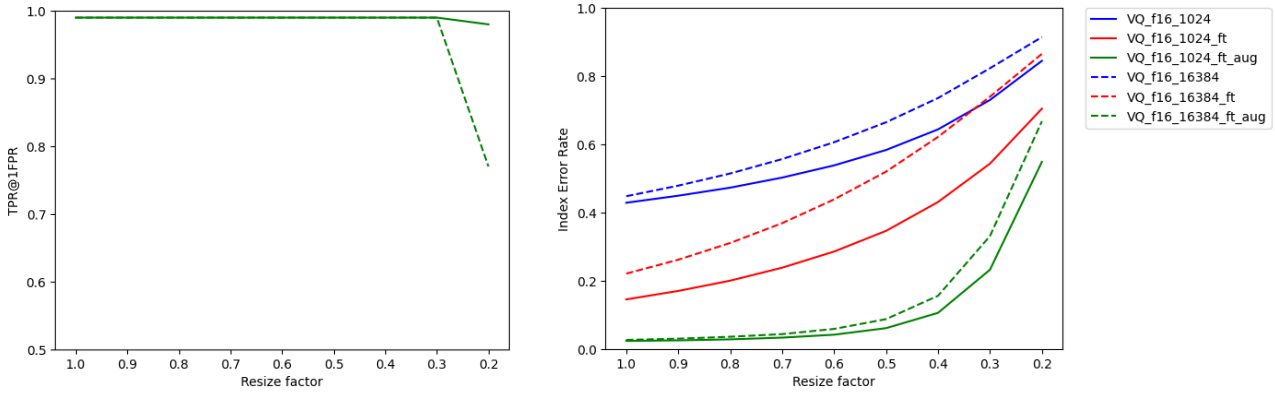


Figure 6: Robustness against Resizing

JPEG compression We compress each image using JPEG compression with a factor in [100 . . . 10]

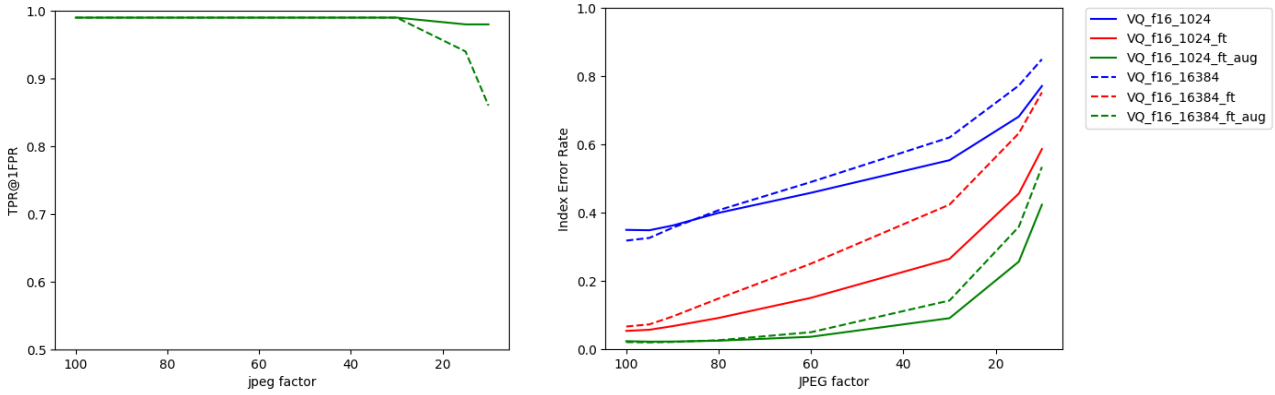


Figure 7: Robustness against JPEG compression

Blur We use Gaussian Blur with a radius parameter in [0 . . . 5].

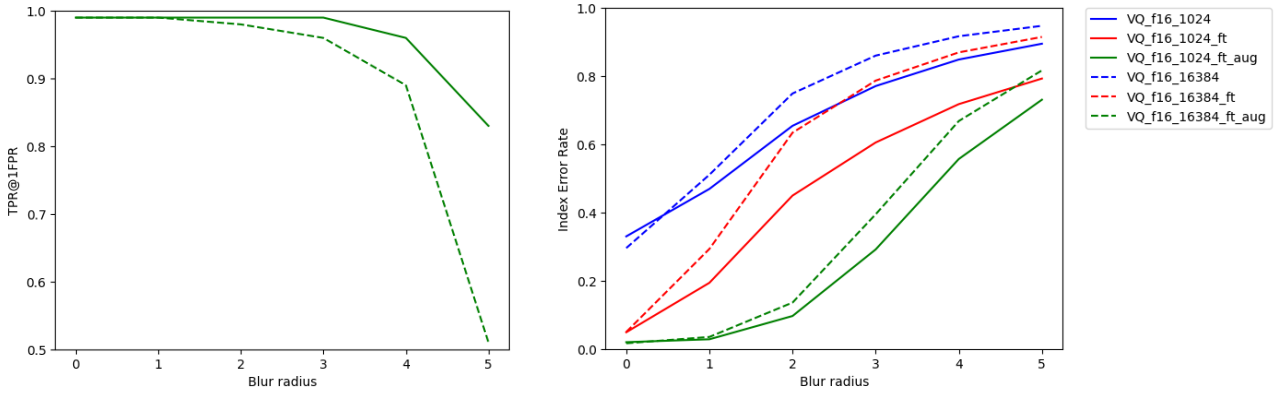


Figure 8: Robustness against Gaussian Blur

Pixelization We pixelate the image by resizing it down using Nearest interpolation by dividing each image dimension by the pixelation strength in [1 . . . 10] and we then resize the image back to it original size using Nearest interpolation again.

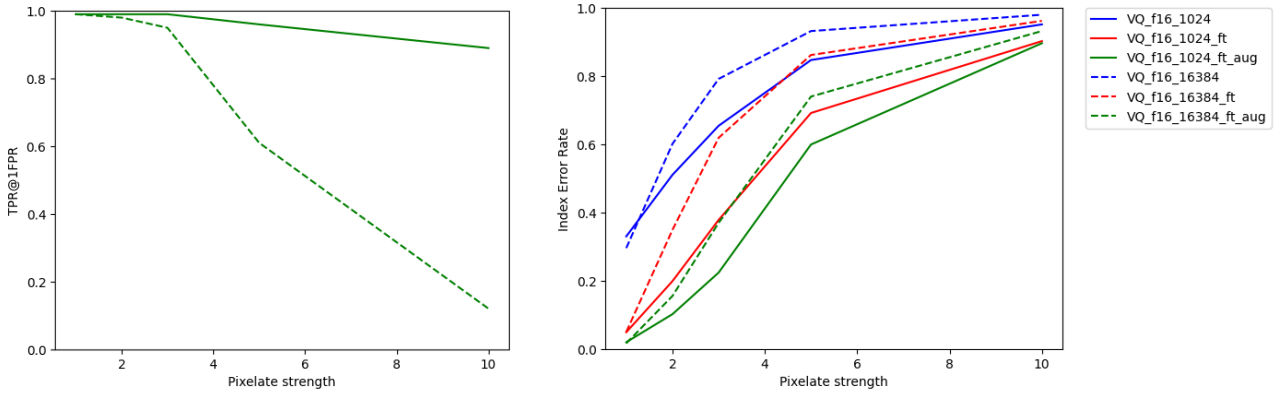


Figure 9: Robustness against Pixelization

Brightness We change the brightness of each image with a factor in [0.5 . . . 1.5]

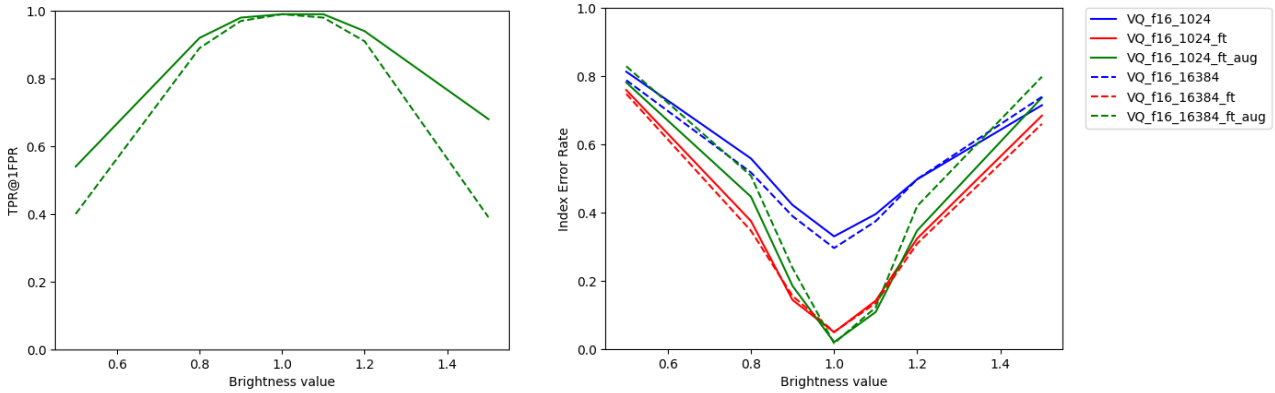


Figure 10: Robustness against Brightness

Crop For cropping we only manage to get the same codes when cropping the image exactly at multiples of compression factors as the encoder is patch based. So removing some patches of the image does not affect the generation of the remaining latent codes, but cropping the image randomly results in a translation of the patches that we encode and therefore in entirely different latent codes.

Acceptable Index Error Rate Our final goal is to be able to tell if the image has been watermarked. The obtained p-value will depend on both the index error rate and the size of the watermarked image.

As an example, let consider that we watermark a 256×256 image with VQ_f16 (with p=0.5 the probability of being in the green list) and after some image transformations we obtain an index error rate at 40%. Assuming the worst case scenario where the wrongly obtained codes are not in

the green list. As the compression factor of the used VQGAN is 16 we get $16 \times 16 = 256$ codes in total, and we obtain a p-value of 1e-3. On average, half the wrong codes are going to be in the green list, given that p=0.5, and we obtain a p-value of 9.3e-23.

4.3.2 Image quality

The results are shown in Table 9 and 10. We observe that the difference between the original image and the watermarked one is quite strong, which is largely due to the reconstruction capabilities of the autoencoder itself, as it has a high 16 compression factor. The image quality between the reconstructed image and the watermarked image give us a better idea on the impact of the watermark we introduce in the quantized latent space. As it is expected the PSNR is low (around 25) as a change in the latent space results in a "semantic" change in pixel space, but the images generally do look similar as it is indicated by a low LPIPS score <0.1.

We also observe that using a bigger codebook, 16384 codes vs 1024 codes, allow us to obtain higher image quality as the nearest vector that we replace in the latent space is closer in terms of l_2 distance to the original vector, at the cost of less robustness as shown in the previous subsection.

Table 9: Image quality metrics for VQGQN_f16_1024_ft_aug

	PSNR↑	SSIM↑	LPIPS↓	SIFID↓
Original-watermarked	20.34	0.52	0.24	0.3
Reconstructed-watermarked	24.91	0.81	0.1	0.07

Table 10: Image quality metrics for VQGQN_f16_16384_ft_aug

	PSNR↑	SSIM↑	LPIPS↓	SIFID↓
Original-watermarked	20.8	0.56	0.20	0.18
Reconstructed-watermarked	26.05	0.84	0.07	0.03

4.3.3 Adversarial attacks robustness

We test the robustness of our approach against black-box adversarial attacks such as square-attack, algo5 table 2 and GSM-DiagonalCMA [74]. We use our testing dataset (100 images from ImageNet) that we watermark using the soft green list approach with a fixed green list and VQGAN_f16_1024 to embed the watermark.

Table 11 shows the obtained success rates of each attack. Compared to the success rates on attacking UnivFD Table 3, where GSM-DiagonalCMA also obtains a 100% success rate, our watermarking approach is a lot more robust toward this kind of attacks without ever considering them during training.

Table 11: Success rate when attacking the watermark detector. All attacks have 10k budget and $l^\infty = 0.03$.

Algorithm	SA	algo5	GSM-DiagonalCMA
Success rate	34.4%	18.0%	34.0%

4.4 Limitations and Extentions

The main limitation of the approach is the weak robustness to geometrical transformations : rotation, stretching, flipping, etc. As the main motivation is to embed a semantic change in the watermarked image, this means that any semantic transformation, as geometrical transformations, would effectively remove the watermark.

An important point that needs to be addressed and verified is the i.i.d hypothesis of the latent codes. We can address the possible correlations by constructing a covariance matrix on a validation dataset and take it into account during the computations of the statistical test. As for the uniformly distributed hypothesis, it should be considered during the estimation of p = probability of being in the green list.

One possible improvement direction of our work in this section would be to try to incorporate a steganography technique that would allow us to store a secret message of length l-bits into the watermarked image, as done in [75], [16], [7]. A naive way to add this to our approach can be to choose the hard green list approach and pick the parity of the code at position i as given in the secret message (0 for odd and 1 for even). The problem with this approach is the lost information due to cropping.

Another interesting practical advantage of embedding the watermark in the latent space is the possibility of adding an additional watermark in the pixel space, as we've shown that the method is robust to small noise introduced in the watermarked images. This would allow us to embed 2 watermarks in a single image.

Our approach is not only limited to images, but it can also be extended to other forms of data such as audio, as it has been tried with the recent chatbot Moshi [76].

5 Combining fake detectors

Passive detectors such as UnivFD can detect AI generated images from a variety of generators but are easily attacked by adversarial attacks, while watermarking approaches work well on watermarked images, but suffer from no-box attacks. The main idea of this section is to combine both passive detectors, such as UnivFD, and active detectors, such as Stable Signature, and attacks detectors introduced in section 3 into one ultimate detector. We then show that our new detector outperforms each single detector on a variety of datasets (real, fake, watermarked, attacked) while being more robust to black-box adversarial and no-box attacks.

5.1 Experiments and results

We use the same train/test dataset as in section 3.1, we consider UnivFD, SA detector section 3.2.2, Log-normal detector and Stable Signature as our detectors.

We consider the output probability of being fake for an input image through the considered detectors as the input to our combining classifier. We try various classifiers, including taking the maximum where we flag the image as fake if a detector flag it as fake, RandomForest, MLP classifier etc. Specific details of combining classifiers can be found in the Appendix Table 15.

To train the combining classifier, we use a train dataset that contain the categories:

- train clean real (not attacked real images, 5k randomly selected from the first 500 classes of ImageNet)
- train clean fake (not attacked fake images, 1k images generated by ProGAN)
- train clean watermarked (not attacked fake watermarked images, 1k from ImageNet watermarked by Stable Signature)
- attacked SA (attacked train clean fake images with SA on UnivFD, 10k budget and $l^\infty = 0.03$)
- attacked LN (attacked train clean fake images with algo5 on UnivFD, 10k budget and $l^\infty = 0.03$)
- attacked IR (modified train clean fake images with no-box ImageRephrase, parameter 0.2)
- attacked DP_Big (modified train clean watermarked images with DiffPure_512x512, parameter 0.2)

For testing, we consider the same categories seen during training, in addition to other categories unseen during training and for which we don't have a dedicated detector:

- clean real : c_r (1k from last 500 classes of ImageNet)
- clean fake : c_f (500 unseen from ProGAN + 500 GuidedDiffusion)
- clean watermarked : c_w (1k unseen watermarked by Stable Signature)
- attacked SA : a_sa (attacked fake images with SA on UnivFD)
- attacked LN : a_ln (attacked fake images with algo5 on UnivFD)
- attacked IR : a_ir (modified fake images with no-box ImageRephrase)
- attacked DP_Big : a_dpb (modified watermarked images with DiffPure_512x512)
- attacked GSM-DiagonalCMA : a_gsmdc (attacked fake images with GSM-DiagonalCMA on UnivFD)
- attacked DiagonalCMA : a_dc (attacked fake images with DiagonalCMA on UnivFD)
- attacked DP : a_dp (modified fake images with no-box DiffPure)

Table 12 presents the accuracy obtained on each test categories by single and combination of detectors. We can observe that combining detectors give better performance overall on all testing dataset. Using LinearSVM as our combining classifier gives the best results across different testing datasets with a slight loss of the accuracy on the clean real and clean fake compared to the vanilla UnivFD.

Table 12: Accuracy on each test dataset of single detectors (first half) and their combination using different classifiers (second half)

Detector Dataset	c_r	c_f	c_w	a_sa	a_ln	a_dpb	a_ir	a_gsmdc	a_dc	a_dp
UnivFD	0.981	0.999	0.208	0.000	0.034	0.140	0.946	0.000	0.035	0.809
LND	0.991	0.073	0.000	0.867	0.984	0.000	0.000	1.000	0.998	0.000
SAD	0.984	0.001	0.000	1.000	0.365	0.000	0.000	0.351	0.457	0.000
IRD	0.993	0.235	0.026	0.001	0.007	0.236	0.996	0.001	0.002	0.987
DPDBig	1.000	0.014	0.979	0.000	0.001	0.997	0.011	0.000	0.002	0.998
Stable Signature	1.000	0.000	1.000	0.000	0.001	0.000	0.000	0.001	0.000	0.000
Nearest Neighbors	0.992	0.893	1.000	1.000	0.996	0.998	0.994	1.000	0.928	1.000
Linear SVM	0.974	0.986	0.999	1.000	0.998	0.998	0.982	1.000	0.998	0.999
RBF SVM	0.988	0.911	1.000	1.000	0.995	0.999	0.997	1.000	0.998	1.000
Gaussian Process	0.994	0.773	1.000	0.999	0.990	0.996	0.991	0.999	0.980	0.999
Decision Tree	0.989	0.831	0.992	0.998	0.990	0.997	0.975	0.997	0.978	0.995
Random Forest	0.989	0.842	0.999	0.998	0.993	0.998	0.984	0.999	0.993	1.000
Neural Net	0.985	0.907	1.000	1.000	0.996	0.999	0.998	1.000	0.998	1.000
AdaBoost	0.989	0.914	1.000	1.000	0.990	0.999	0.991	1.000	0.981	1.000
Naive Bayes	0.951	0.980	1.000	1.000	0.998	1.000	0.998	1.000	0.998	1.000
QDA	0.958	0.982	1.000	1.000	0.999	1.000	0.998	1.000	0.998	1.000
CombiningMax	0.951	0.999	1.000	1.000	0.999	0.999	0.998	1.000	0.998	1.000

5.2 Can we attack a detector of an attack ?

As the name of this subsection suggests we test to see if combining detectors actually allow us to gain robustness toward adversarial attacks.

We attack the Combining Linear SVM from the previous subsection which combines UnivFD, SA detector, LN detector, DP_Big detector, IR_detector and Stable Signature. We use SA, algo5 from table 2 and GSM-DiagonalCMA as our black-box attacks. And we use the clean fake images from our test set as our dataset.

Figure 11 shows the success rates of the attacks as a function of the iterations. For comparison, we also report the success rates against the vanilla UnivFD in Figure 12.

We observe that we can still successfully attack the combination of detectors on some images even if we include a detector of the used attack in our combination, but the attacks take significantly more queries to succeed compared to attacking UnivFD alone. And, the attacks achieve a far lower success rate when attacking the combination of detectors.

This shows that we do gain robustness toward an attack by training a detector to detect it, in addition to some unseen attacks. We can also train a new detector to detect the attacks on the combination of the detectors and integrate it, which in theory should make the new combination even more robust. But the question remains: how many detectors should we add before making the attack completely unsuccessful (0% success rate) ?

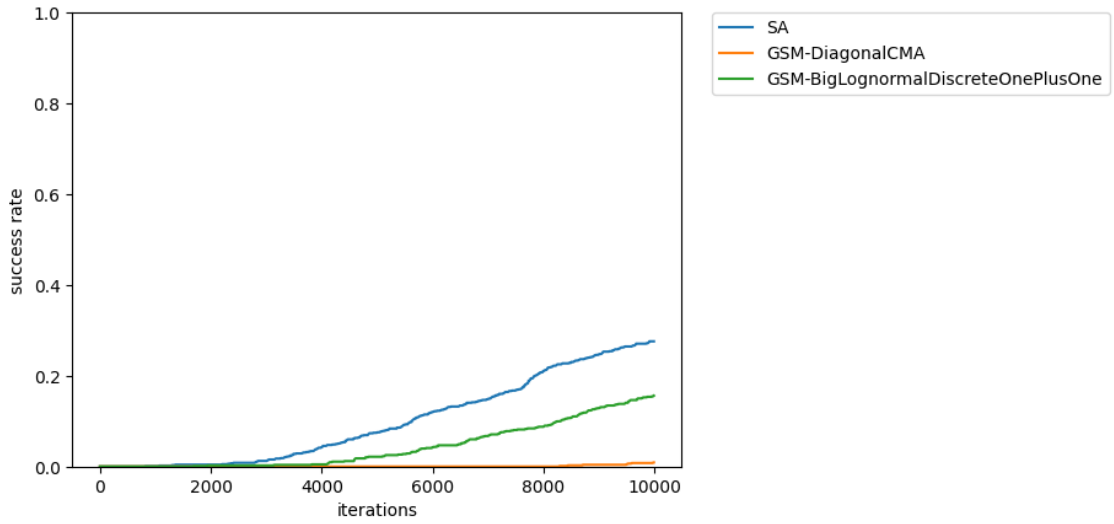


Figure 11: Success rate progression with the number of queries on combining Linear SVM.

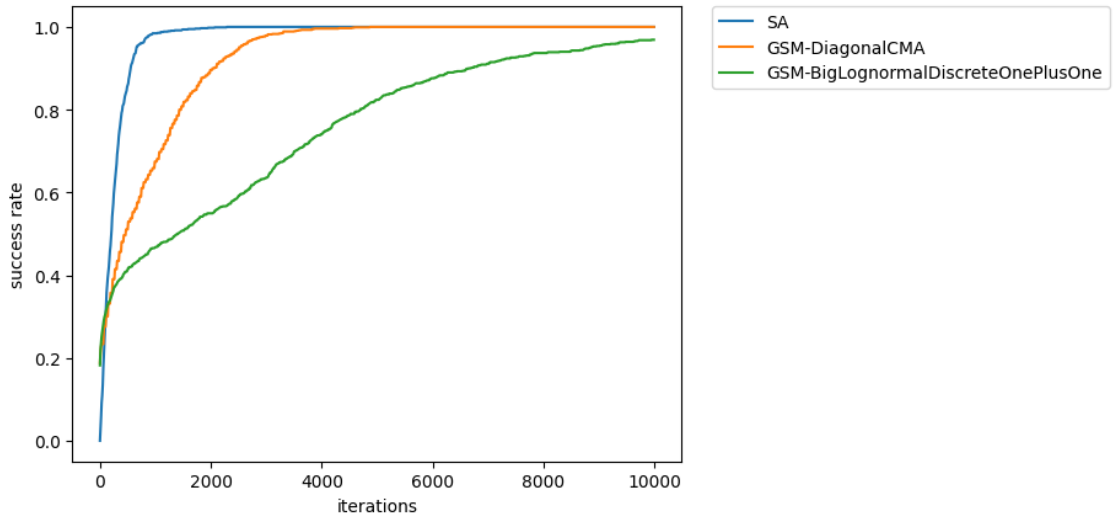


Figure 12: Success rate progression with the number of queries on UnivFD.

6 Conclusion

We demonstrate that we can train specific detectors for no-box and black-box attacks. Our findings indicate that while combining multiple fake and attack detectors may not provide perfect protection, it can significantly enhance the robustness of our system against various types of attacks. By making the adversary's job more difficult, we can effectively mitigate the risks associated with these types of attacks and improve the overall security of our system.

We propose a new approach to watermarking images through biasing their latent codes generated by VQGAN this leads to output images very similar to their reconstructed version while being robust to various non-geometrical transformations.

References

[1] Danielle Citron Robert Chesney. *Deepfakes and the New Disinformation War*. https://www.foreignaffairs.com/articles/world/2018-12-11/deepfakes-and-new-disinformation-war. [Online; accessed 19-July-2024]. 2018.

[2] TODD C. HELMUS. *Artificial Intelligence, Deepfakes, and Disinformation*. https://www.rand.org/content/dam/rand/pubs/perspectives/PEA1000/PEA1043-1/RAND_PEA1043-1.pdf. [Online; accessed 19-July-2024]. 2022.

[3] Bertin Martens. *The European Union AI Act: premature or precocious regulation?* https://www.bruegel.org/analysis/european-union-ai-act-premature-or-precocious-regulation. [Online; accessed 03-August-2024]. 2024.

[4] Tambiama Madiega. *Generative AI and watermarking*. https://www.europarl.europa.eu/RegData/etudes/BRIE/2023/757583/EPRS_BRI(2023)757583_EN.pdf. [Online; accessed 03-August-2024]. 2023.

[5] Sheng-Yu Wang et al. "CNN-generated images are surprisingly easy to spot... for now". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 8695–8704.

[6] Miki Tanaka and Hitoshi Kiya. "Fake-image detection with Robust Hashing". In: *2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech)*. IEEE. 2021, pp. 40–43.

[7] J Zhu. "HiDDeN: hiding data with deep networks". In: *arXiv preprint arXiv:1807.09937* (2018).

[8] Ahmad M Nagm, Mohamed Torky, and Khaled Y Youssef. "A novel watermarking approach for protecting image integrity based on a hybrid security technique". In: *arXiv preprint arXiv:2110.08777* (2021).

[9] Joana C Costa et al. "How deep learning sees the world: A survey on adversarial attacks & defenses". In: *IEEE Access* (2024).

[10] Maksym Andriushchenko et al. "Square Attack: A Query-Efficient Black-Box Adversarial Attack via Random Search". In: *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIII*. Ed. by Andrea Vedaldi et al. Vol. 12368. Lecture Notes in Computer Science. Springer, 2020, pp. 484–501. DOI: 10.1007/978-3-030-58592-1_29. URL: https://doi.org/10.1007/978-3-030-58592-1%5C_29.

[11] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2574–2582.

[12] Chuan Guo, Jared S Frank, and Kilian Q Weinberger. "Low frequency adversarial perturbation". In: *arXiv preprint arXiv:1809.08758* (2018).

[13] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. "Hopskipjumpattack: A query-efficient decision-based attack". In: *2020 ieee symposium on security and privacy (sp)*. IEEE. 2020, pp. 1277–1294.

[14] Mehrdad Saberi et al. "Robustness of ai-image detectors: Fundamental limits and practical attacks". In: *arXiv preprint arXiv:2310.00076* (2023).

[15] Utkarsh Ojha, Yuheng Li, and Yong Jae Lee. "Towards Universal Fake Image Detectors that Generalize Across Generative Models". In: *CVPR*. 2023.

[16] Pierre Fernandez et al. "The stable signature: Rooting watermarks in latent diffusion models". In: *arXiv preprint arXiv:2303.15435* (2023).

[17] Weili Nie et al. "Diffusion models for adversarial purification". In: *arXiv preprint arXiv:2205.07460* (2022).

[18] Ismail Labiad et al. "Log-normal Mutations and their Use in Detecting Surreptitious Fake Images". In: *arXiv preprint arXiv:2409.15119* (2024).
[19] Patrick Esser, Robin Rombach, and Bjorn Ommer. "Taming transformers for high-resolution image synthesis". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 12873–12883.
[20] Shruti Agarwal and Hany Farid. "Photo forensics from JPEG dimples". In: *2017 IEEE workshop on information forensics and security (WIFS)*. IEEE. 2017, pp. 1–6.
[21] Zahra Moghaddasi, Hamid A Jalab, and Rafidah Md Noor. "Image splicing detection using singular value decomposition". In: *Proceedings of the second international conference on internet of things, data and cloud computing*. 2017, pp. 1–5.
[22] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.
[23] Mingjian Zhu et al. "Gendet: Towards good generalizations for ai-generated image detection". In: *arXiv preprint arXiv:2312.08880* (2023).
[24] Peng Zhou et al. "Learning rich features for image manipulation detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1053–1061.
[25] Nan Zhong et al. "Patchcraft: Exploring texture patch for efficient ai-generated image detection". In: *arXiv preprint arXiv:2311.12397* (2024), pp. 1–18.
[26] Shilin Yan et al. "A Sanity Check for AI-generated Image Detection". In: *arXiv preprint arXiv:2406.19435* (2024).
[27] Ingemar Cox et al. *Digital watermarking and steganography*. Morgan kaufmann, 2007.
[28] Zhicheng Ni et al. "Reversible data hiding". In: *IEEE Transactions on circuits and systems for video technology* 16.3 (2006), pp. 354–362.
[29] Nikos Nikolaidis and Ioannis Pitas. "Robust image watermarking in the spatial domain". In: *Signal processing* 66.3 (1998), pp. 385–403.
[30] Matthieu Urvoy, Dalila Goudia, and Florent Autrusseau. "Perceptual DFT watermarking with improved detection and robustness to geometrical distortions". In: *IEEE Transactions on Information Forensics and Security* 9.7 (2014), pp. 1108–1119.
[31] Junlin Ouyang et al. "Color image watermarking based on quaternion Fourier transform and improved uniform log-polar mapping". In: *Computers & Electrical Engineering* 46 (2015), pp. 419–432.
[32] Adrian G Bors and Ioannis Pitas. "Image watermarking using DCT domain constraints". In: *Proceedings of 3rd IEEE International Conference on Image Processing*. Vol. 3. IEEE. 1996, pp. 231–234.
[33] Zhen Li, Kim-Hui Yap, and Bai-Ying Lei. "A new blind robust image watermarking scheme in SVD-DCT composite domain". In: *2011 18th IEEE International Conference on Image Processing*. IEEE. 2011, pp. 2757–2760.
[34] Xiang-Gen Xia, Charles G Boncelet, and Gonzalo R Arce. "Wavelet transform based watermark for digital images". In: *Optics Express* 3.12 (1998), pp. 497–511.
[35] Liu Ping Feng, Liang Bin Zheng, and Peng Cao. "A DWT-DCT based blind watermarking algorithm for copyright protection". In: *2010 3rd international conference on computer science and information technology*. Vol. 7. IEEE. 2010, pp. 455–458.
[36] Haribabu Kandi, Deepak Mishra, and Subrahmanyam RK Sai Gorthi. "Exploring the learning capabilities of convolutional neural networks for robust image watermarking". In: *Computers & Security* 65 (2017), pp. 247–268.
[37] Jae-Eun Lee, Young-Ho Seo, and Dong-Wook Kim. "Convolutional neural network-based digital image watermarking adaptive to the resolution of image and watermark". In: *Applied Sciences* 10.19 (2020), p. 6854.

[38] Xiyang Luo et al. "Distortion agnostic deep watermarking". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 13548–13557.
[39] Honglei Zhang et al. "Robust data hiding using inverse gradient attention". In: *arXiv preprint arXiv:2011.10850* (2020).
[40] Bingyang Wen and Sergul Aydore. "Romark: A robust watermarking system using adversarial training". In: *arXiv preprint arXiv:1910.01221* (2019).
[41] Zihan Wang et al. "Data hiding with deep learning: A survey unifying digital watermarking and steganography". In: *IEEE Transactions on Computational Social Systems* 10.6 (2023), pp. 2985–2999.
[42] Pin-Yu Chen et al. "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models". In: *Proceedings of the 10th ACM workshop on artificial intelligence and security*. 2017, pp. 15–26.
[43] Arjun Nitin Bhagoji et al. "Practical black-box attacks on deep neural networks using efficient query mechanisms". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 154–169.
[44] Chun-Chen Tu et al. "Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 742–749.
[45] Moustafa Alzantot et al. "Genattack: Practical black-box attacks with gradient-free optimization". In: *Proceedings of the genetic and evolutionary computation conference*. 2019, pp. 1111–1119.
[46] Qizhang Li, Yiwen Guo, and Hao Chen. "Practical no-box adversarial attacks against dnns". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 12849–12860.
[47] Johannes Ballé et al. "Variational image compression with a scale hyperprior". In: *arXiv preprint arXiv:1802.01436* (2018).
[48] Zhengxue Cheng et al. "Learned image compression with discretized gaussian mixture likelihoods and attention modules". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 7939–7948.
[49] Robin Rombach et al. "High-resolution image synthesis with latent diffusion models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.
[50] Prafulla Dhariwal and Alexander Nichol. "Diffusion models beat gans on image synthesis". In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.
[51] Xuandong Zhao et al. "Invisible image watermarks are provably removable using generative ai". In: *arXiv preprint arXiv:2306.01953* (2023).
[52] Siddhant Bhambri et al. "A survey of black-box adversarial attacks on computer vision models". In: *arXiv preprint arXiv:1912.01667* (2019).
[53] Nicolas Papernot et al. "Distillation as a defense to adversarial perturbations against deep neural networks". In: *2016 IEEE symposium on security and privacy (SP)*. IEEE. 2016, pp. 582–597.
[54] Chuan Guo et al. "Countering adversarial images using input transformations". In: *arXiv preprint arXiv:1711.00117* (2017).
[55] Leonid I Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.
[56] Fangzhou Liao et al. "Defense against adversarial attacks using high-level representation guided denoiser". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1778–1787.
[57] Samangouei Pouya. "Defense-GAN: protecting classifiers against adversarial attacks using generative models". In: *Retrieved from https://arXiv: 1805.06605* (2018).

[58] Florian Tramèr et al. “Ensemble adversarial training: Attacks and defenses”. In: *arXiv preprint arXiv:1705.07204* (2017).
[59] Aleksander Mądry et al. “Towards deep learning models resistant to adversarial attacks”. In: *stat* 1050.9 (2017).
[60] Dongyu Meng and Hao Chen. “Magnet: a two-pronged defense against adversarial examples”. In: *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 2017, pp. 135–147.
[61] Bin Liang et al. “Detecting adversarial image examples in deep neural networks with adaptive noise reduction”. In: *IEEE Transactions on Dependable and Secure Computing* 18.1 (2018), pp. 72–85.
[62] Paula Harder et al. “Spectraldefense: Detecting adversarial attacks on cnns in the fourier domain”. In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–8.
[63] W Xu. “Feature squeezing: Detecting adversarial exa mples in deep neural networks”. In: *arXiv preprint arXiv:1704.01155* (2017).
[64] Jeremy Rapin and Olivier Teytaud. *Nevergrad - A gradient-free optimization platform*. https://GitHub.com/FacebookResearch/Nevergrad. 2018.
[65] Mehrdad Saberi et al. “Robustness of AI-Image Detectors: Fundamental Limits and Practical Attacks”. In: *ICLR*. 2024.
[66] Mehdi Boroumand, Mo Chen, and Jessica Fridrich. “Deep residual network for steganalysis of digital images”. In: *IEEE Transactions on Information Forensics and Security* 14.5 (2018), pp. 1181–1193.
[67] Riccardo Corvi et al. “On the detection of synthetic images generated by diffusion models”. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.
[68] Zhendong Wang et al. “Dire for diffusion-generated image detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 22445–22455.
[69] Patrick Grommelt et al. “Fake or JPEG? Revealing Common Biases in Generated Image Detection Datasets”. In: *arXiv preprint arXiv:2403.17608* (2024).
[70] John Kirchenbauer et al. “A watermark for large language models”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 17061–17084.
[71] Aiwei Liu et al. “A survey of text watermarking in the era of large language models”. In: *ACM Computing Surveys* (2024).
[72] Matthew Tancik, Ben Mildenhall, and Ren Ng. “Stegastamp: Invisible hyperlinks in physical photographs”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2117–2126.
[73] Yuxin Wen et al. “Tree-rings watermarks: Invisible fingerprints for diffusion images”. In: *Advances in Neural Information Processing Systems* 36 (2024).
[74] Youhei Akimoto and Nikolaus Hansen. “Diagonal acceleration for covariance matrix adaptation evolution strategies”. In: *Evolutionary computation* 28.3 (2020), pp. 405–435.
[75] Tu Bui et al. “Rosteals: Robust steganography using autoencoder latent space”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 933–942.
[76] Alexandre Défossez et al. “Moshi: a speech-text foundation model for real-time dialogue”. In: *arXiv preprint arXiv:2410.00037* (2024).

Appendix

A Modifiers of algorithms

A.1 Modifiers dedicated to tensors

When the Smooth operator is applied to a black-box optimization method in Nevergrad, periodically, it tries to replace the current best candidate x by $Smooth(x)$. If $Smooth(x)$ has a better loss value than x, then x is replaced by $Smooth(x)$. $Smooth(x)$ is defined as a tensor with the same shape as x, with $Smooth(x)_i$ defined as x_i with probability 75%, and, otherwise, the average of the x_j for j the points at distance at most 1 of i in the indices of the tensor x. Smooth means that this tentative smoothing is tested once per 55 iterations, SuperSmooth once per 9 iterations; see Table 13.

Table 13: Parametrization of the Smooth operator, which operates on a $s \times s$-window.

Parameter	Value
Frequency of update (per iteration)	$1/2$ (ZetaSmooth) $1/3$ (UltraSmooth) $1/9$ (SuperSmooth) $1/55$ (Default smooth)
Size of smoothing window s	$s = 3$

A.2 Modified dedicated to adversarial attacks

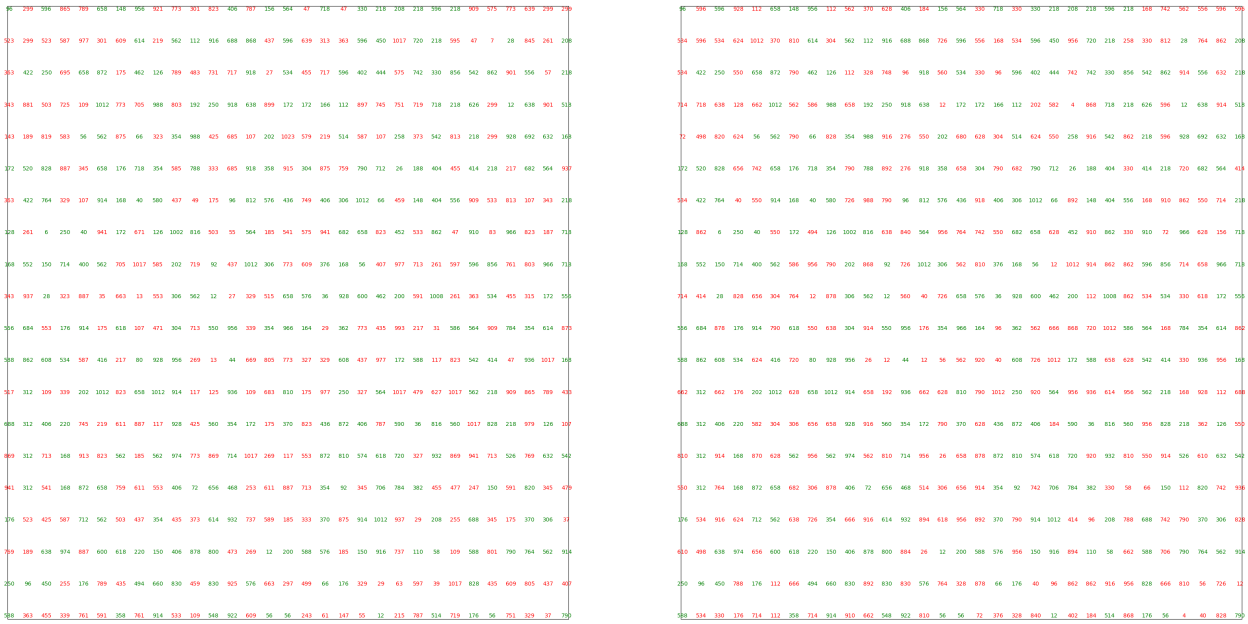
The modifiers G and GSM used for arrays work as follows:

$$\begin{aligned} loss_G(x) &= loss(0.03 \times sign(x)) \\ loss_{GSM}(x) &= loss(0.03 \times sign(convolve(x,k))) \end{aligned}$$

in the context of an image with values in $[0,1]$ and an amplitude 0.03 in l^∞. k refers to a Gaussian kernel of width $r/8$ with r the width of the image.

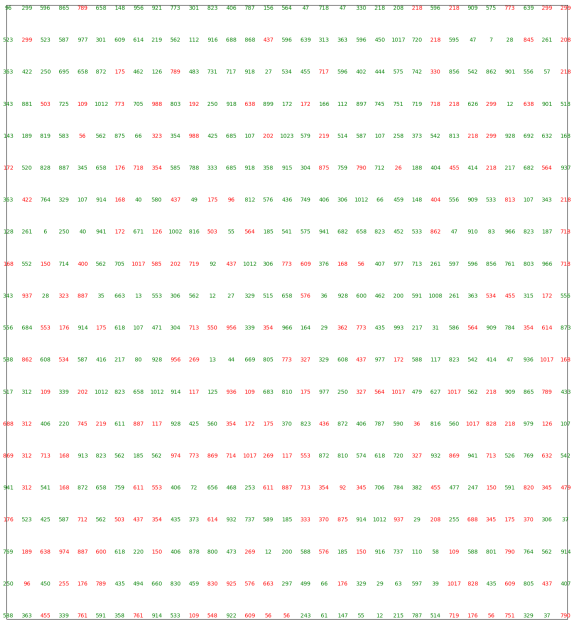
B Watermarking Examples

B.1 Latent codes

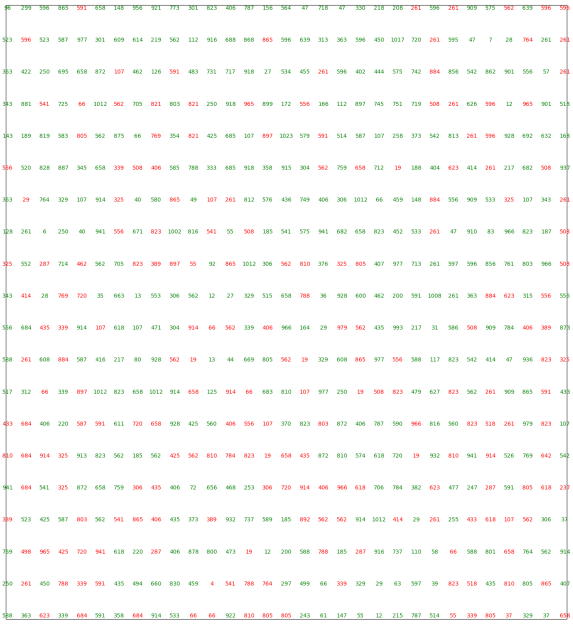


Original codes watermarked codes

Figure 13: Left: codes obtained from encoding the input image. Right: watermarked codes. The green color represents that the codes have not changed, while the red color represents that the codes have been modified



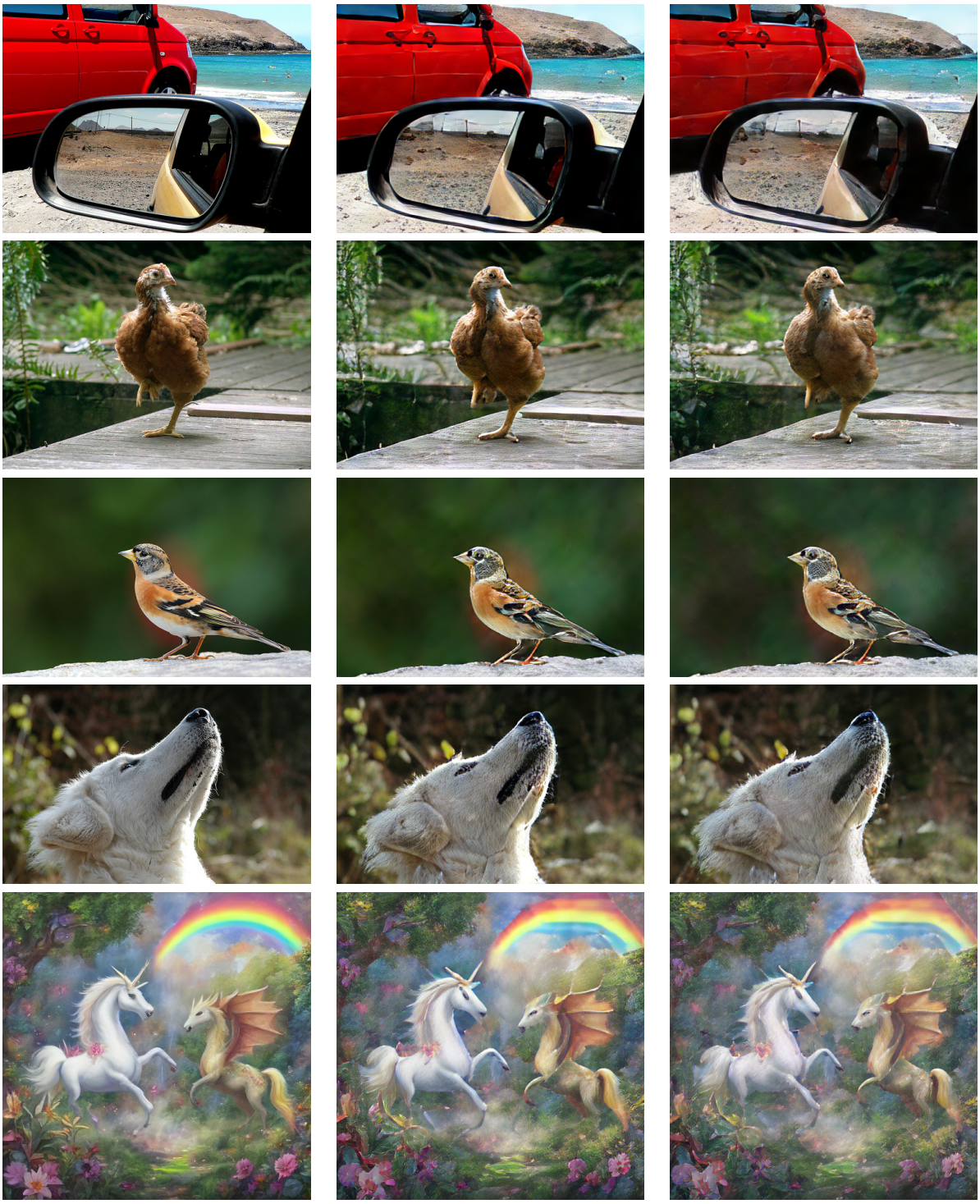
Original codes



watermarked codes

Figure 14: Left: codes obtained from encoding the input image. Right: watermarked codes. The green color represents that the codes have not changed, while the red color represents that the codes have been modified

B.2 Example watermarked images



Original | Reconstructed | Watermarked

Figure 15: Examples of original, reconstructed and watermarked images, using VQGAN_f16_1024_ft_aug and soft green list approach

C Parameters of the learning model

Table 14: Hyperparameters of our learning model. We work on Dataset2 with each purifier (Dataset2-DP and Dataset2-IR) with parameter 0.1, which is split in 80% train, 10% test, 10% validation.

Hyperparameter	Value
Type of model	SRnet
learning rate	10^{-4}
Number of epochs	20
Hardware	1 GPU
Training time	$\simeq$ 1 hour
Data Augmentation	JPEG 0.2 prob at 96 quality Resize((256, 256)) RandomHorizontalFlip(0.5) RandomCrop((224, 224))

D Combining Classifiers

Table 15: Implementation parameters of the tested Combining classifiers.

Name	**Implementation details**
Nearest Neighbors	KNeighborsClassifier(3)
Linear SVM	SVC(kernel="linear", C=0.2, random_state=42)
RBF SVM	SVC(gamma=2, C=1, random_state=42)
Gaussian Process	GaussianProcessClassifier(1.0 * RBF(1.0), random_state=42)
Decision Tree	DecisionTreeClassifier(max_depth=10, random_state=42)
Random Forest	RandomForestClassifier(max_depth=20, n_estimators=30,max_features=10, random_state=42)
Neural Net	MLPClassifier(hidden_layer_sizes=np.array([100, 100]), alpha=0.001, learning_rate_init=1e-2, batch_size=64, max_iter=10, random_state=42)
AdaBoost	AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=10), algorithm="SAMME", random_state=42)
Naive Bayes	GaussianNB()
QDA	QuadraticDiscriminantAnalysis()
CombiningMax	takes the maximum of the fake probabilities