*entropy*

MDPI

*Article*

# Are Classification Deep Neural Networks Good for Blind Image Watermarking?

**Vedran Vukotić [1], Vivien Chappelier [1] and Teddy Furon [2,*]**

[1] Lamark, 35000 Rennes, France; vedran.vukotic@imatag.com (V.V.); vivien.chappelier@lamark.fr (V.C.)
[2] INRIA, CNRS, IRISA, University of Rennes, 35000 Rennes, France
* Correspondence: teddy.furon@inria.fr

Received: 6 December 2019; Accepted: 4 February 2020; Published: 8 February 2020

**Abstract:** Image watermarking is usually decomposed into three steps: (i) a feature vector is extracted from an image; (ii) it is modified to embed the watermark; (iii) and it is projected back into the image space while avoiding the creation of visual artefacts. This feature extraction is usually based on a classical image representation given by the Discrete Wavelet Transform or the Discrete Cosine Transform for instance. These transformations require very accurate synchronisation between the embedding and the detection and usually rely on various registration mechanisms for that purpose. This paper investigates a new family of transformation based on Deep Neural Networks trained with supervision for a classification task. Motivations come from the Computer Vision literature, which has demonstrated the robustness of these features against light geometric distortions. Also, adversarial sample literature provides means to implement the inverse transform needed in the third step above mentioned. As far as zero-bit watermarking is concerned, this paper shows that this approach is feasible as it yields a good quality of the watermarked images and an intrinsic robustness. We also tests more advanced tools from Computer Vision such as aggregation schemes with weak geometry and retraining with a dataset augmented with classical image processing attacks.

**Keywords:** digital watermarking; Deep Learning; feature extraction

## 1. Introduction

Deep Learning (DL) has completely revolutionized the field of Computer Vision. It started with image classification [1] where a DL network is trained to recognize classes of images and it is now spreading to any task of Computer Vision—object recognition [2], instance search [3], localization [4], similar image retrieval [5,6].

The transfer property is underlying this versatility—a DL network trained on some supervised task (e.g., classification) has been shown to perform well on other applications, even non-supervised tasks like similarity search.

A DL network is comprised of several layers. Supervised training learns the best parameters of each layer in order to minimize a loss function counting the prediction errors over an annotated image set. While the last layer of a classifier network provides a vector of probabilities that the input image belongs to the classes, the internal auxiliary data (the output of the previous layers) also happen to contain very relevant information about the input image. Other tasks can indeed tap these data. Many Computer Vision works take a trained classification network off the shelf, keep the first layers as is and only re-train the deepest layers for their specific task [5].

Following this trend, state-of-the-art image search algorithms tap the output of an internal layer and use them to derive a global descriptor of the image [5,6]. This way, finding similar images boils down to finding close vectors in a Euclidean high dimensional space, where efficient fast search engines exist. It has been shown that these global descriptors are compact and discriminative while at the same time robust to valuemetric (e.g., JPEG, noise, filtering) and light geometric (e.g., cropping, rotation, scaling) distortions [7].

## *1.1. Problem Formulation*

This paper investigates whether this approach is also suitable for zero-bit watermarking, with the hope of benefiting from this apparent robustness. The layers of a DL network play the role of the extraction function yielding a vector to be watermarked. This raises the following challenges:

- How to invert this highly non-linear extraction function? Once the feature vector is watermarked, how to map it back into the image space to create an image whose feature vector equals a given target vector?
- How to take into account a perceptual model ensuring the quality of the watermarked image?
- How to guarantee a required false positive level as there is no probabilistic modeling of these features?
- Which network architecture provides the best extraction function? Is it possible to robustify this extraction by fine-tuning the underlying network?

## *1.2. Prior Works*

The main trend dealing with DL and watermarking is actually the protection of DL networks. It aims at proving the ownership of a network by embedding a watermark into the learned parameters of the model [8,9].

The connection between machine learning and *image* watermarking is surprisingly old. References [10,11] protect images with a classical watermarking technique and a neural network is only used at the decoding side in place of a maximum likelihood decoder based on a statistical model, which may not be accurate. J. Stolarek and P. Lipiński learn a transformation [12] as proposed in this paper. However, this transform is dedicated to one specific host image. References [13,14] have a similar position than ours in the sense that they use a single network at the detection side. Reference [14] needs to train another network for embedding, whereas [13] uses back-propagation as we do (see Section 4.3). There are also shortcomings that watermarking experts would point out. These are multi-bit watermarking schemes that fail guaranteeing a probability of false alarm. Indeed, Reference [14] briefly looks at turning their multi-bit embedding into a zero-bit scheme. They report a detection rate lower than 10% for a JPEG compression with quality factor $Q \leq 90$ (see [14], Figure 11-combined). There is no secret key preventing renewability when compromised. There is no simple control of the embedding distortion. The embedding distortion reported in both papers is indeed quite high with an average PSNR around 35 dB . They are not robust to geometric transformations (unless a registration is performed first [13]).

As a related work, we mention Reference [15] which makes a very good comparison between attacks in watermarking and attacks on DL networks (*a.k.a.* adversarial sample—see Section 3.2).

This article is the journal version of the conference paper [16] proposing improvements to robustify the feature vector extraction. This is achieved by using aggregation schemes with weak geometry (RMAC and WELDON—see Section 3.1) and by an adversarial retraining of the network (see Section 4.4). This last idea is coming from References [13,14].

The contribution of the study is to assess the relevance of Deep Neural Networks as a feature extraction when watermarking requirements are prioritized: embedding and detection use a secret key which is easily renewed if compromised, the embedding distortion is under control, the probability of false alarm is guaranteed and the test images are large.

## *1.3. Structure of the Paper*

Section 2 presents the state-of-the-art in zero-bit watermarking when combined with a linear extraction function. Section 3 presents elements of DL networks with a focus on adversarial examples. Section 4 details our new watermarking scheme and a re-training process to increase the robustness by strengthening the invariance of the network. Section 5 reports our experimental work following by a discussion in Section 6.

## 2. Zero-Bit Watermarking

Our scheme belongs to the TEMIT approach (a wording coined by Ton Kalker.): Transform, EMbed, Inverse Transform. We denote by $\mathsf{Tr}(\cdot)$ the transformation extracting a feature vector, so-called the host signal $\mathbf{x}_o = \mathsf{Tr}(\mathcal{I}_o)$ of dimension $n$, from an image $\mathcal{I}_o$. The embedding modifies it into $\mathbf{x}_m = \mathsf{Emb}(\mathbf{x}_o)$. This function implicitly depends on a secret key. For simplicity, the paper focuses on a zero-bit watermarking scheme [17–19]: we hide the presence of a secret mark, not a message. The watermarked image is given by $\mathcal{I}_m = \mathcal{I}_o + \mathsf{Tr}^{-1}(\mathbf{x}_m - \mathbf{x}_o)$, where $\mathsf{Tr}^{-1}(\cdot)$ is the inverse transform function.

At the detection side, the image under scrutiny is $\mathcal{I}_a$ with $\mathbf{x}_a = \mathsf{Tr}(\mathcal{I}_a)$. It is deemed as a watermarked image if $\mathbf{x}_a$ belongs to the acceptance region $\mathcal{D} \subset \mathbb{R}^n$. The use of a secret key is implicit in these notations.

### 2.1. The Hypercone Detector

This paper focuses on one specific zero-bit watermarking scheme: the hypercone detector [17]. This scheme is provably good [19] and the detector is blind and oblivious to the watermark, host and noise powers.

The acceptance region $\mathcal{D}$ is a dual hypercone of axis $\mathbf{a} \in \mathbb{R}^n$ ($\|\mathbf{a}\| = 1$) and half angle $0 < \theta < \pi/2$ defined as:

$$\mathcal{D} := \{\mathbf{x} \in \mathbb{R}^n : |\mathbf{x}^\top \mathbf{a}| > \|\mathbf{x}\| \cos(\theta)\}. \tag{1}$$

Vector $\mathbf{a}$ indeed plays the role of the secret key.

We now define the following function $R(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$

$$\mathbf{x} \to R(\mathbf{x}) = (\mathbf{x}^\top \mathbf{a})^2 \rho(\theta) - \|\mathbf{x}\|^2, \tag{2}$$

with $\rho(\theta) := (\tan^2(\theta) + 1)$. This quantity is negative when $\mathbf{x} \notin \mathcal{D}$, null when $\mathbf{x}$ lies on the boundary of $\mathcal{D}$ and positive inside the dual hypercone. Indeed, when $R(\mathbf{x}) > 0$, this quantity is the amount of power of Gaussian white noise to be added onto $\mathbf{x}$ in order to push it outside the hypercone with high probability [19].

This gives a rationale for watermarking vectors at the embedding side. We push the host vector $\mathbf{x}_o$ to a location $\mathbf{x}_m$ deep inside $\mathcal{D}$ s.t. $R(\mathbf{x}_m)$ is as big as possible. This provably increases the robustness against a white Gaussian noise addition in the feature space.

### 2.2. Linear and Invertible Extraction Function

The hypercone detector is at the core of some image watermarking techniques [18] where the extraction function is (almost) a Parseval tight frame $T$ (e.g., selection of some DCT or DWT coefficients). Once the embedding modifies $\mathbf{x}_o$ in $\mathbf{x}_m$, the watermarked image is obtained as

$$\mathcal{I}_m = \mathcal{I}_o + T^\top(\mathbf{x}_m - \mathbf{x}_o). \tag{3}$$

A constraint on the image Euclidean distance $\|\mathcal{I}_m - \mathcal{I}_o\|_2 \leq C$ (e.g., expressed in terms of MSE or PSNR) is ensured if $\|\mathbf{x}_m - \mathbf{x}_o\|_2 \leq C$ because Parseval frame preserves Euclidean distance. In the feature domain, the embedding then amounts to solve the following problem:

$$\mathbf{x}_m = \arg \max_{\mathbf{x} : \|\mathbf{x} - \mathbf{x}_o\|_2 \leq C} R(\mathbf{x}). \tag{4}$$

It is known that $\mathbf{x}_m$ belongs to the 2D-hyperplane containing the origin $\mathbf{x}_o$ and the axis of the hypercone $\mathbf{a}$ [17,20], so that (4) boils down to a line search over angle $\beta \in [0, \pi/2]$ defining:

$$\mathbf{x} = \mathbf{x}_o + C\cos(\beta)\mathbf{u} - C\sin(\beta)\mathbf{v}, \tag{5}$$

with $(\mathbf{u}, \mathbf{v})$ a basis of this hyperplane:

$$\mathbf{u} := \mathsf{sign}(\mathbf{x}_o^\top \mathbf{a})\mathbf{a}, \quad \mathbf{v} := \frac{\mathbf{x}_o - (\mathbf{x}_o^\top \mathbf{u})\mathbf{u}}{\|\mathbf{x}_o - (\mathbf{x}_o^\top \mathbf{u})\mathbf{u}\|}. \tag{6}$$

Vectors $\mathbf{x}_o$ and $\mathbf{a}$ are statistically independent, so that for large $n$, $\mathbf{x}_o^\top \mathbf{a} \approx 0$. In that case, the close form solution is [19]:

$$\beta^\star = \arccos\sqrt{1 - \|\mathbf{x}_o\|^2 C^{-2}\cos^4\theta}, \tag{7}$$

if $C \geq \|\mathbf{x}_o\|\cos^2\theta$, otherwise $\mathbf{x}_o$ cannot be watermarked (i.e., pushed into region $\mathcal{D}$) because $C$ is too small.

The embedding is thus given by a close form solution or a simple line search over $\beta$. This is thanks to an invertible extraction function $\mathsf{Tr}(\cdot)$ preserving the Euclidean distance.

## 3. Deep Learning Feature Extraction

This section considers a neural network used as an image classifier over $n_c$ classes. It is denoted by function $\mathsf{F}$ that computes a vector from an image: $\mathbf{p} = \mathsf{F}(\mathcal{I}) \in [0,1]^{n_c}$. Once the parameters of the classifier have been learned from a training set, the network predicts classes of images. The output $\mathbf{p}$ is a vector of estimated probabilities $p_k = \hat{P}(k|\mathcal{I})$ that content $\mathcal{I}$ belongs to class $k$. In the end, the class of the query image is given by

$$\mathsf{C}(\mathcal{I}) = \arg\max_{1\leq k\leq n_c} p_k. \tag{8}$$

### 3.1. Architecture

The function $\mathsf{F}$ is decomposed as $\mathsf{F} = \mathsf{S} \circ \mathsf{N}$, where $\mathsf{S}$ is the softmax function ensuring that $\mathbf{p}$ is a probability distribution (i.e., $\sum_{k=1}^{n_c} p_k = 1$ and $0 \leq p_k \leq 1, \forall k \in \{1, \dots, n_c\}$). Function $\mathsf{N}$ is the network per se, a composition of $\ell_C$ convolutional layers and $\ell_F$ fully connected layers.

$$\mathsf{N} = \mathsf{L}^F_{\ell_F} \circ \mathsf{L}^F_{\ell_F-1} \circ \dots \mathsf{L}^F_1 \circ \mathsf{V} \circ \mathsf{L}^C_{\ell_C} \circ \mathsf{L}^C_{\ell_C-1} \circ \dots \mathsf{L}^C_1. \tag{9}$$

Each layer is a non-linear process as it encompasses an activation function such as the well-known $\mathsf{ReLU}$. Function $\mathsf{V}$ is a layer flattening the 3D activation map of the last convolutional layer into a vector suitable for the first fully connected layer.

The convolutional layers output data living in very high-dimensional spaces that are not practical for direct representation. The output of a convolutional layer is a three-dimensional tensor of size $W \times H \times K$, where $K$ is the number of feature maps and $(H, W)$ are the dimensions of each feature map. The total dimensionality is in an order of tens or hundreds of thousands, especially in higher layers of deeper network architectures. Our first approach watermarks the output of one layer. If it is a convolutional layer, the flattening function $\mathsf{V}$ produces a vector. It is used directly or by performing a dimensionality reduction [16]. In this work, a second approach considers a non-linear pooling strategy. It aggregates the 3D tensor in a specific way for the retention of localization information. It has been shown that geometry even weakly preserved improves a lot the transferability of a classification network to tasks like image search [21]. This work evaluates two aggregation methods in the context of watermarking with neural architectures—RMAC [21] and WELDON [22]. That aggregation is not part of the classification network (9). It is just added after a convolutional layer for the purpose of watermarking only.

RMAC, namely Regional Maximum Activation of Convolutions, is a multi-scale aggregation method. It retains locality information by utilizing region-specific feature vectors for multiple, overlapping, multi-scale regions. It was developed primarily for content based image retrieval (CBIR) applications of convolutional neural networks. Let $\chi_i(\mathbf{c})$ represent the response of feature $i \in [K]$ at coordinate $\mathbf{c} \in [W] \times [H]$ (Notation $[n]$ means the set of integers $\{1, 2, \dots, n\}$.). Let $f_{\mathcal{R},i} = \max_{\mathbf{c} \in \mathcal{R}} \chi_i(\mathbf{c})$ be the maximum activation of the $i^{th}$ feature over a valid spatial region $\mathcal{R} \subset [W] \times [H]$. A region feature vector is then defined as $\mathbf{f}_{\mathcal{R}} = \begin{bmatrix} f_{\mathcal{R},1} & \dots & f_{\mathcal{R},i} & \dots & f_{\mathcal{R},K} \end{bmatrix}^T$. Then $\ell_2$ normalization and PCA-whitening is applied to each region feature vector. The global (for the whole image $\mathcal{I}$) RMAC descriptor is the sum of the region feature vectors followed by a final $\ell_2$ normalization. Its dimension is thus $K$.

$$\mathsf{E}_{\text{RMAC}}(\mathcal{I}) \propto \left( \sum_{\mathcal{R}} W_{\mathcal{R}}^{-1/2} \mathbf{f}_{\mathcal{R}} / \|\mathbf{f}_{\mathcal{R}}\| \right) \quad \text{s.t.} \quad \|\mathsf{E}_{\text{RMAC}}(\mathcal{I})\| = 1. \tag{10}$$

As RMAC regions are defined at different scales, the first level consists of the whole feature map. The second level consists of 4 regions. The third level consists of 9 regions where each overlap at least 40% with the regions of the previous level and so on. The regions of the first 3 levels of RMAC aggregation are shown, for a feature map of size $14 \times 14 \times n$ in Figure 1. In the experimental section, we used an RMAC layer consisting of 5 scale levels. A higher number of scales would not be possible given the size of the feature maps and smaller number of levels were less performing.

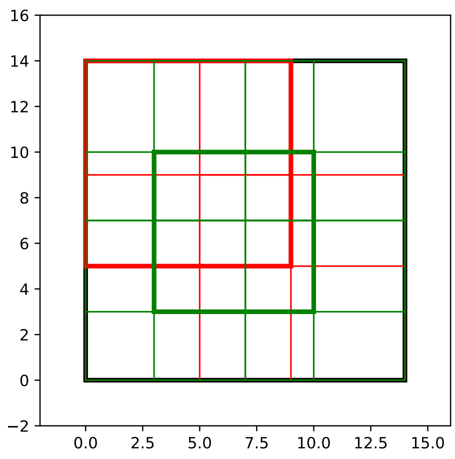

**Figure 1.** Three Regional Maximum Activation of Convolution (RMAC) levels and their corresponding pooling regions for a feature map of size $14 \times 14 \times n$. The first level, denoted in black and bolded, consist of only one region of size $14 \times 14$. The second level is denoted in red and consists of 4 regions (one of which is bolded for emphasis) of size $9 \times 9$. The third level, denoted in green, consists of 9 regions (one of which is again bolded for emphasis) of size $7 \times 7$ that are offset so that they have at least 40% overlap with the regions of the previous layer.

WELDON, namely WEakly supervised Learning of Deep cOnvolutional neural Networks, is another method of regional pooling incorporating negative evidences. For a given feature $i \in [K]$, it looks for the $k$ coordinates having the largest activation. This is written as:

$$s_i^{\text{top}} = \sum_{j=1}^{k} \chi_i(\mathbf{c}_j), \tag{11}$$

where coordinates $\{\mathbf{c}_j\} \subset [W] \times [H]$ are ordered s.t. $\chi_i(\mathbf{c}_1) > \chi_i(\mathbf{c}_2) > \ldots > \chi_i(\mathbf{c}_{WH})$. In the same way, it sums the $m$ lowest responses:

$$s_i^{\text{low}} = \sum_{j=1}^{m} \chi_i(\mathbf{c}_{WH+1-j}). \tag{12}$$

The authors state that introducing this kind of negative instances (regions that best supports the absence of a class) improves the avoidance of local minima and improves the overall classification results. The final descriptor for image $\mathcal{I}$ has size $K$ and is obtained by summing the two regional scores:

$$\mathsf{E}_{\text{WELDON}}(\mathcal{I}) = \mathbf{s}^{\text{top}} + \mathbf{s}^{\text{low}}. \tag{13}$$

### 3.2. Adversarial Samples

Adversarial sampling is a recent trend showing that DL classifiers can easily be fooled [23,24]. The attack forges an image $\mathcal{I}_a$ visually similar to an original image $\mathcal{I}_o$ but with a different prediction $\mathsf{C}(\mathcal{I}_a) \neq \mathsf{C}(\mathcal{I}_o)$.

In a targeted attack, a specific class is given, say $k_t$, together with an original image $\mathcal{I}_o$ not belonging to this class. The attack finds $\mathcal{I}_a$ as close as possible to $\mathcal{I}_o$ s.t. $\mathsf{C}(\mathcal{I}_a) = k_t$:

$$\mathcal{I}_a = \arg \min_{\mathcal{I}:\mathsf{C}(\mathcal{I})=k_t} \mathsf{Dist}(\mathcal{I}, \mathcal{I}_o), \tag{14}$$

with Dist a measure of distortion (usually, the $L_\ell$-norm of $(\mathcal{I} - \mathcal{I}_o)$ with $\ell \in \{0, 1, 2, +\infty\}$). In a white box scenario, the attacker knows the network architecture and parameters.

Problem (14) is impossible to be solved as the region of images classified as $k_t$ is unknown. It is replaced by

$$\begin{aligned} \mathcal{I}_a &= \arg\min \mathsf{J}_\lambda(\mathcal{I}) && (15) \\ \mathsf{J}_\lambda(\mathcal{I}) &:= \mathsf{Loss}(\mathcal{I}, k_t) + \lambda.\mathsf{Dist}(\mathcal{I}, \mathcal{I}_o). && (16) \end{aligned}$$

where $\lambda \in \mathbb{R}^+$ and Loss is a loss function measuring how far $\mathcal{I}$ is from being classified as class $k_t$. For instance, the loss is a divergence (e.g., Cross-Entropy) between the predicted probability distribution $\mathsf{F}(\mathcal{I})$ and the targeted one-shot vector $\mathbf{p}^\star$, i.e., $p_k^\star = \delta_{k_t}(k)$, or more simply:

$$\mathsf{Loss}(\mathcal{I}, k_t) = |\max_{k \neq k_t}(\mathsf{F}(\mathcal{I})_k) - \mathsf{F}(\mathcal{I})_{k_t}|_+, \tag{17}$$

with $|a|_+ = a$ if $a > 0$ and 0 otherwise. This way

$$\begin{aligned} \mathsf{C}(\mathcal{I}) = k_t &\Leftrightarrow \mathsf{Loss}(\mathcal{I}, k_t) = 0, && (18) \\ \mathsf{C}(\mathcal{I}_o) \neq k_t &\Leftrightarrow \mathsf{Loss}(\mathcal{I}_o, k_t) > 0. && (19) \end{aligned}$$

### 3.3. Practical Solutions

The minimization problem (15) implicitly uses the domain of images of the same size $h \times l$ and same number of channels $c$ as the original image $\mathcal{I}$, say $\{0, 1, \ldots, 255\}^{h \times l \times c}$. A relaxation looks for a solution over the continuous set $[0, 255]^{h \times l \times c}$. Quantization onto integers is applied on the continuous solution to obtain an image. This box-constrained optimization often uses a differentiable mapper $\mathsf{m}$ which is a monotonic bijection from $\mathbb{R}$ to $[0, 255]$, e.g., $\mathsf{m}(x) = 255(\tanh(x) + 1)/2$ as in Reference [24]. This changes the box-constrained optimization problem (14) into an unconstrained minimization of $\mathsf{J}_\lambda(\mathsf{m}(X))$ over $X \in \mathbb{R}^{h \times l \times c}$.

Practical attacks are mostly based on a gradient descent [25,26]. The computation of the gradient of the objective function is not so difficult because one can back-propagate the computation of the gradient through the layers of the network. The attack initializes a random starting point $X^{(0)} = \mathsf{m}^{-1}(\mathcal{I}_o)$ and iteratively computes

$$X^{(i+1)} = X^{(i)} - \eta \nabla \mathsf{J}_\lambda(\mathsf{m}(X^{(i)})). \tag{20}$$

It stops when the objective function no longer decreases substantially. This gradient descent goes down to a local minimum when converging. Several rounds with different initialization are competing. Advanced numerical solvers have been also used in the literature, for example, ADAM [24]. The main difficulty is to set $\lambda$ to a well chosen constant. Theoretically, there exists a range of $\lambda$ values where the solution of (15) coincides with the solution of (14). In practice, this range is unknown and we apply a line search over $\lambda$ on top of the gradient descent.

Simpler attacks start with a given distortion budget, $\mathsf{Dist}(\mathcal{I}_a, \mathcal{I}_o) \leq C$, and set $\lambda = 0$. The gradient descent starts at the original image, that is, $X^{(0)} = \mathsf{m}^{-1}(\mathcal{I}_o)$ and iterates until either $\mathsf{C}(\mathsf{m}(X^{(j)})) = k_t$ or $\mathsf{Dist}(\mathsf{m}(X^{(j)}), \mathcal{I}_o) > C$. In the latter case, the attack failed finding an adversarial sample within the distortion budget.

## 4. Application to Zero-Bit Watermarking

The key idea of this paper is (i) to use the network $\mathsf{N}$ or part of it (i.e., the first layers) as the extraction function $\mathsf{Tr}(\cdot)$, (ii) to modify the extracted vector as explained in Section 2, (iii) to use an adversarial sample mechanism of Section 3 as $\mathsf{Tr}^{-1}(\cdot)$ in order to put back the marked vector into the image. This raises several issues cleared in this section.

### *4.1. Need of a Locality Transform*

Our proposed system can function with any neural architecture. Denote by $\mathsf{E}$ the set of the first layers and the aggregation (be it the flattening function, RMAC, or WELDON) used for extracting an embedding of an image: $\mathbf{e} = \mathsf{E}(\mathcal{I}) \in \mathbb{R}^K$. Yet, the representational coordinate system of the embedding space does not depict the feasible locations (i.e., the typical values of these extracted vectors). A neural network never provides a zero-mean isotropic embedding $\mathbf{e}$. The usual culprit being the asymmetrical activation functions, such as ReLU, that have a $\mathbb{R}_{\geq 0}$ codomain.

This rises the need for a mapping from the original coordinate system to a local coordinate system. We empirically model a local coordinate system by providing images to the DL network and analyzing their extracted feature vectors in the representation space. We use two possible mappings:

- **Centering**: $\mathbf{x} = \mathsf{L}(\mathbf{e}) = \mathbf{e} - \bar{\mathbf{e}}$,
- **PCA**: $\mathbf{x} = \mathsf{L}(\mathbf{e}) = \bar{\Sigma}^{-1/2}(\mathbf{e} - \bar{\mathbf{e}})$,

where $\bar{\mathbf{e}}$ is the empirical mean and $\bar{\Sigma}$ is the empirical covariance matrix. The first option preserves the dimensionality $n = K$, while the second may operate a reduction by keeping the $n < K$ biggest eigenvalues of $\bar{\Sigma}$. In the end, $\mathsf{Tr} = \mathsf{L} \circ \mathsf{E}$.

### *4.2. False Alarm Probability*

The false alarm probability is usually defined as $P_{\mathsf{fa}} := \mathbb{P}(\mathbf{X} \in \mathcal{D})$ where the random vector $\mathbf{X}$ models the feature vector of an original image. For the hypercone detector, under the weak assumption that $\mathbf{X}$ has an isotropic distribution (e.g., a Gaussian white distribution), this probability has a close form expression:

$$P_{\mathsf{fa}} = \mathsf{I}_{\cos^2(\theta)}((n-1)/2, 1/2), \tag{21}$$

where $\mathsf{I}$ is the regularized Beta incomplete function.

This assumption does not hold for feature vectors extracted by the network. Even the empirical centering and whitening doesn't guarantee a distribution *exactly* isotropic.

Instead, we prefer modifying the definition of the probability of false alarm. For a given image under scrutiny, the extracted features are seen as a fixed vector. It is the acceptance region which is random: the axis direction of the hypercone is a random vector $\mathbf{A}$ uniformly distributed on the unit hypersphere. It happens that this probability of false alarm has the exact same expression as (21). For a required false positive level, (21) is inverted so as to find the corresponding half angle value $\theta$.

Another use of this equation is to compute the $p$-value: for given $\mathbf{x}$ and $\mathbf{a}$, the $p$-value is the probability (21) computed for a cosine equalling $\cos(\theta) = |\mathbf{x}^\top \mathbf{a}|/\|\mathbf{x}\|$. It means that if we were drawing $O(1/p)$ random secret keys $\mathbf{A}$, on expectation, only one of them would give a normalized correlation bigger or equal than that $\cos(\theta)$. The smaller the $p$-value is, the more convinced we are that $\mathbf{a}$ was the secret key used at the embedding.

### *4.3. The Objective Function and Imposed Constraints*

A bad idea is to stick too closely to the watermarking described in Section 2.2. It amounts to first find $\mathbf{x}_m$ (4) and then to use the back propagation to craft an image whose feature vector is as close as possible to $\mathbf{x}_m$.

Watermarking in the feature space and crafting the watermarked image are better done jointly by defining:

$$\begin{aligned} \mathcal{I}_w &= \arg \min_{\mathsf{Dist}(\mathcal{I},\mathcal{I}_o) \leq C} \mathsf{Loss}(\mathcal{I}) && (22) \\ \mathsf{Loss}(\mathcal{I}) &= -\mathsf{R}(\mathsf{Tr}(\mathcal{I})), && (23) \end{aligned}$$

and letting an adversarial sample mechanism solving this optimization problem. The distortion function Dist is for instance the Mean Square Error (expressed in dB as a PSNR). Note that with these notations, a watermark is detected in image $\mathcal{I}$ if $\mathsf{Loss}(\mathcal{I}) < 0$.

While minimizing the loss and thus iteratively generating the watermarked image, additional constraints can be applied. For instance, the watermark can be attenuated selectively in the spatial domain by a perceptual mask to make it less perceivable. We use the Structural Similarity SSIM heatmap for that purpose. The watermarking is then performed by a simple gradient descent (see Algorithm 1).

### *4.4. Improving Robustness by Retraining*

The introduction explains our motivation of benefiting of the inherent robustness of known classification DL networks in the very same way as it happened in image search. Retraining aims at increasing this robustness against typical attacks in watermarking applications. These are *image to image* transformations like translation, rotation, cropping or JPEG compression. Therefore building an augmented dataset is made easy by applying these transformations on the original training set. Note that watermarking is kept out of the retraining process. Our goal is to strengthen the invariance of the original classification network without any bias to the feature extraction method for watermarking. Later on, watermarking will naturally inherit this invariance.

The network is trained from scratch on a series of attacks of increasing difficulty. This is driven by the classification loss on the original network architecture (without the watermark feature extraction based on aggregation and whitening). For training, the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) dataset [27] was used. It consists of 1.2 million images of 1000 classes. Attacks to the images were applied dynamically through a data augmentation pipeline making attacks of increasing difficulty as an input to the network (elaborated in Section 5.5).

**Algorithm 1:** Proposed watermarking algorithm.

```
Data: input image $\mathcal{I}_o$, watermarking signal $\mathbf{a}$
Result: watermarked image $\mathcal{I}_w$
$i \leftarrow 0$;
$\mathcal{I}_i \leftarrow \mathcal{I}_o$;
repeat
    /* obtain embedding                                          */
    $\mathbf{x} \leftarrow \mathsf{Tr}(\mathcal{I}_i)$;
    $\mathsf{Loss}(\mathcal{I}_i) \leftarrow -R(\mathbf{x})$;

    /* perform one step update                                   */
    $\mathbf{g} \leftarrow \nabla \mathsf{Loss}(\mathcal{I}_i)$;
    $\mathcal{I}_{i+1} \leftarrow \mathcal{I}_i - \eta \mathbf{g}$;

    if performing SSIM heatmap attenuation then
        $\mathbf{h}_{ssim} \leftarrow SSIM_{heatmap}(\mathcal{I}_o, \mathcal{I}_{i+1})$;
        /* attenuate watermark with psycho-visual heatmap        */
        $\mathcal{I}_\delta \leftarrow \mathcal{I}_{i+1} - \mathcal{I}_o$;
        $\mathcal{I}_{i+1} \leftarrow \mathcal{I}_o + \mathbf{h}_{ssim} \otimes \mathcal{I}_\delta$;
    end
    /* compute indicator values                                  */
    Compute $\mathsf{Dist}(\mathcal{I}_o, \mathcal{I}_{i+1})$

    $i \leftarrow i + 1$;
until stopping criterion met;
$\mathcal{I}_w \leftarrow \mathcal{I}_i$;
```

## 5. Experiments

The watermarking system is both architecture and layer agnostic, meaning that any layer can a priori be used for watermarking. This paper considers one specific architecture at different layers, exploring their goodness for watermarking by evaluating robustness and perceptual quality.

### 5.1. Experimental Protocol

The experimental part evaluates our proposed system based on the off-the-shelf *VGG19* [28] convolutional neural network pretrained on *ImageNet* (available at https://keras.io/applications/#vgg19). We perform all the experiments on photos from the test set P "professional" provided as part of the CLIC challenge [29]. This dataset consists of 117 professionally taken photos. The typical size is $2048 \times 1350$ pixels.

### 5.2. Image Quality

When the stopping criterion is expressed in terms of PSNR, the quality of the watermarked image is roughly acceptable above 40 dB. Figure 2 illustrates this with the reference Lenna image on its top row. The watermark signal in the image space looks like uniform noise when watermarking lower layers and gets more structured when processing deeper layers, as shown in Figure 3.

A psycho-visual attenuation as specified in Algorithm 1 improves the quality of the image. Our experiments used the structural similarity (SSIM) heatmap, but any other psycho-visual model (e.g., Butteraugli [30]) can be integrated transparently. Figure 2 shows that the watermark is less perceivable on the right column thanks to attenuation in flat regions (skin, hat, wall) for a given $PSNR$ target. This almost does not hurt robustness $\mathsf{R}(\mathsf{Tr}(\mathcal{I}_w))$.

The downside is the increased number of steps (and thus time) necessary to obtain a watermark of the same robustness. Figure 4 shows a 3D representation of the watermarking process. The first two axis corresponds to the projection of $\mathrm{Tr}(\mathcal{I}_i)$ onto the basis $(\mathbf{u}, \mathbf{v})$ of Equation (6), while the third component accounts for the energy remaining in the complimentary space. The stopping criterion was met in 10 or 13 iterations, respectively without or with perceptual attenuation.

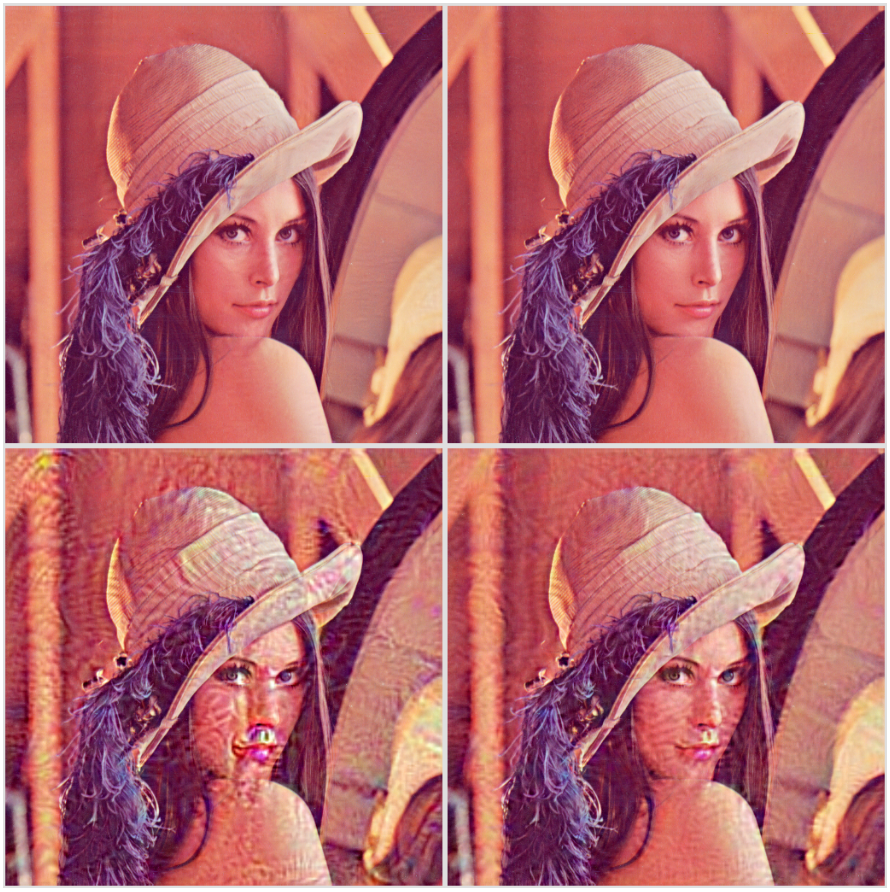

**Figure 2.** Top: $PSNR = 42$ dB and $P_{fa} = 10^{-8}$ without (left) and with (right) structural similarity attenuation. Bottom: 'exaggerated' watermark at $PSNR = 25$ dB without (left) and with (right) structural similarity attenuation.

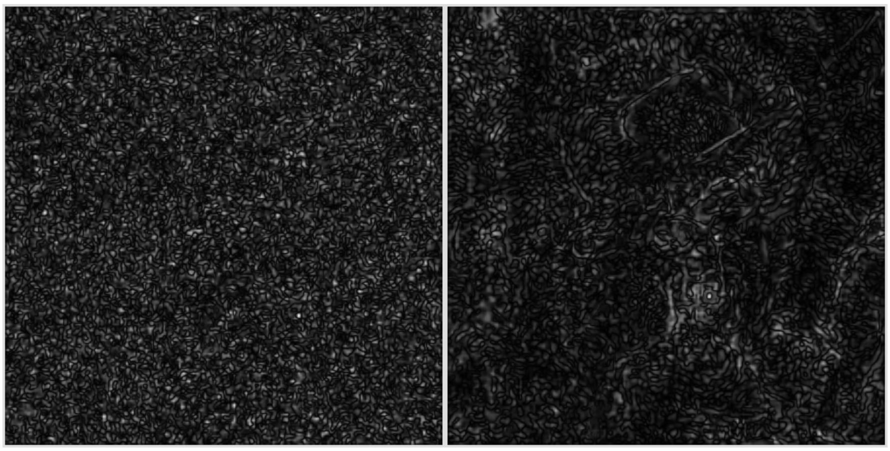

**Figure 3.** The watermark signal in the image space. Left - *block1_pool*: last pooling layer of the first convolutional block. Right - *fc1*: last fully-connected layer.

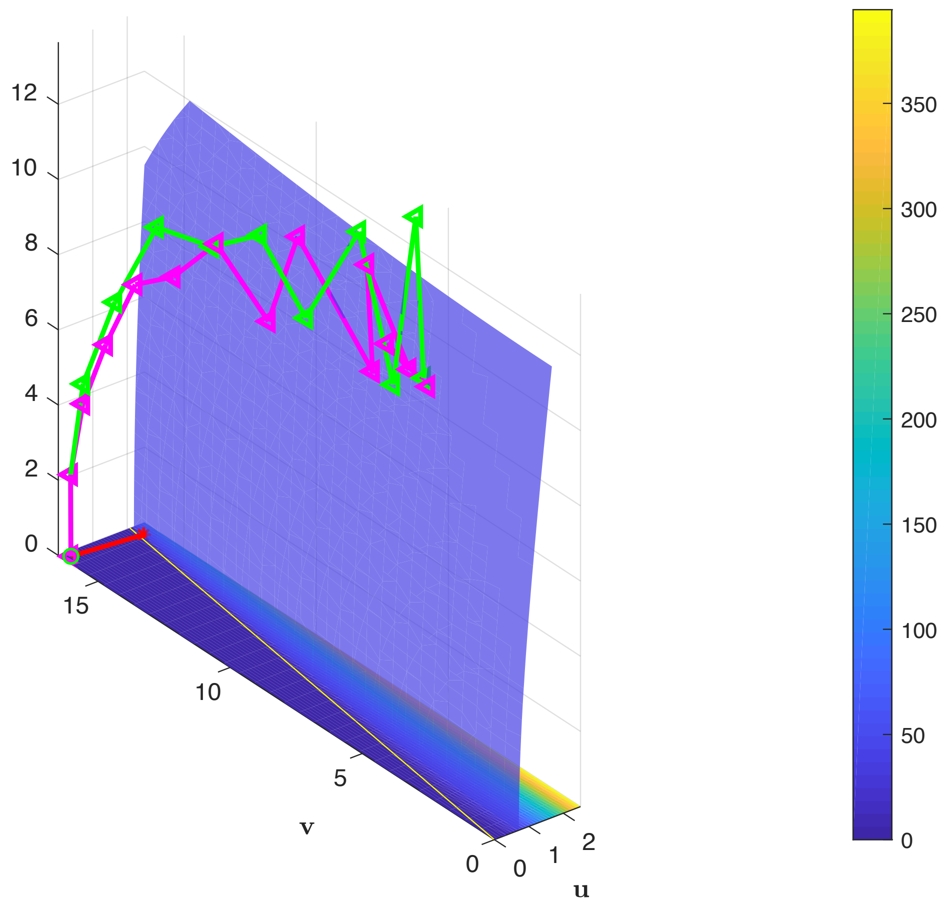

**Figure 4.** 3D view of the iterations without (green) and with (magenta) the perceptual attenuation. The 3D shape shows the set $\{\mathbf{x} : \mathsf{R}(\mathbf{x}) = c\}$. The red line shows the embedding of Section 2.2 providing the same robustness. The yellow line is the boundary of the hypercone. The heatmap of $\mathsf{R}(\mathbf{x})$ on plane $(\mathbf{u}, \mathbf{v})$.

## *5.3. From One Layer to Another*

Table 1 gives the $\log_{10}$ $p$-value for the watermarked image and after a rotation of 5°. For the moment, no aggregation scheme is used. The first part of the table illustrates the behaviour when centering is used as a locality transform, thus retaining the original dimensionality. The second part of the table illustrates the behaviour with PCA with a reduction to 256 dimensions.

In general, watermarking at deeper layers offers more robustness. Each convolutional–pooling block adding some invariance to rotation and scaling, the best layer is indeed the first fully-connected layer.

**Table 1.** Watermarking with different VGG19 layers with the same target $P_{fa} = 1\times 10^{-6}$ and a stopping criterion of $PSNR \leq 42$ dB. ($pool[i]$ being the output of the $i^{th}$ pooling layer and $fc[i]$ being the output of the $i^{th}$ fully-connected layer of the VGG19 architecture).

| **Layer Name** | **Emb. Dim.** | **Average $\log_{10}$ $p$-Value Marked Image** | **Average $\log_{10}$ $p$-Value Rotation 5°** |
|---|---|---|---|
| Original dimensionality, centering | | | |
| pool1 | 802 816 | −64.49 | −0.68 |
| pool2 | 401 408 | −58.32 | −1.13 |
| pool3 | 200 704 | −52.25 | −3.80 |
| pool4 | 100 352 | −21.31 | −3.81 |
| pool5 | 25 088 | −4.95 | −2.24 |
| fc1 | 4 096 | −5.27 | −4.41 |
| fc2 | 4 096 | −4.11 | −1.70 |
| PCA to 256 dimensions | | | |
| pool1 | 802 816 | −3.65 | −0.93 |
| pool2 | 401 408 | −12.30 | −1.36 |
| pool3 | 200 704 | −8.13 | −3.20 |
| pool4 | 100 352 | −8.19 | −2.42 |
| pool5 | 25 088 | −4.61 | −2.41 |
| fc1 | 4 096 | −5.61 | −3.20 |
| fc2 | 4 096 | −5.22 | −1.69 |

## *5.4. Robustness without Aggregation*

We evaluate the robustness of the first fully-connected *fc1* layer of VGG19 at the expense of a lower dimensionality ($n = 4096$). From now on, the stopping criterion is defined as $PSNR = 42$ dB and the probability of false alarm is fixed to $P_{\mathsf{fa}} = 10^{-3}$. To define a baseline, we watermark the same images with the linear wavelet transform as described in Section 2.2 with the same distortion constraint.

Figure 5 illustrates the robustness against rotation. Wavelets drop to a detection rate of around 25% after a rotation of only 0.5° and it ceases to detect watermarks after a rotation of 1°. On contrary, the robustness of our technique smoothly degrades with the strength of the attack.

Figure 6 illustrates the robustness against cropping. The performances again smoothly degrade with the strength of the attack: 50% of watermark detection when cropped to 90% of their original content and continues to detect some watermarks down to a crop of 50%. The system outperforms the classic approach. This is not surprising as DWT was not designed for geometric invariance. Yet, it shows that accurate synchronization is of utmost importance with linear transforms (e.g., DCT, DWT, random projection), whereas it is far less stringent in the new approach.

On top of this, we observe that the system is robust to horizontal flips! We were not expecting this feature. After investigations, we provide the following explanation. Since horizontal flips are naturally and artificially occurring in the training dataset of ImageNet, the neural network seems to have learned a transform invariant to this.

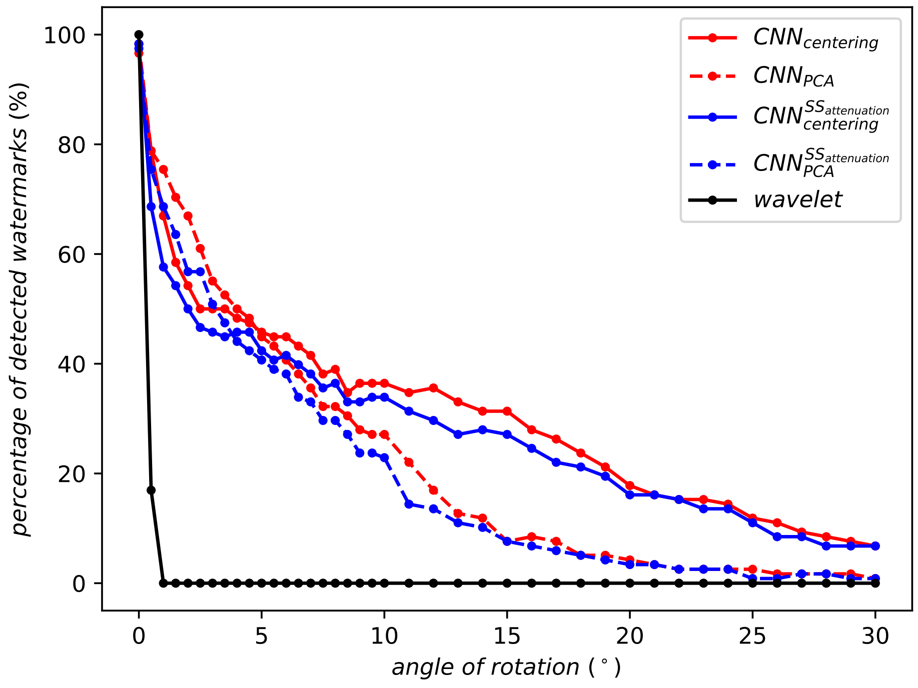

**Figure 5.** Robustness against rotation.

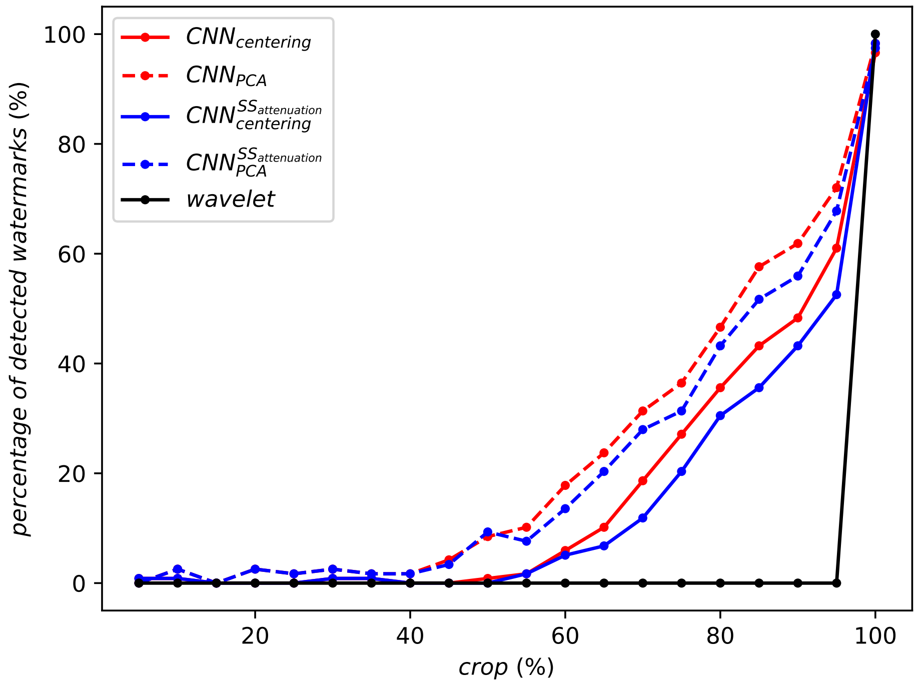

**Figure 6.** Robustness against cropping.

Figure 7 illustrates the robustness against JPEG compression. DWT performs better and we argue that this is caused by the lower dimensionality ($n = 4k$ *vs.* $1M$). The rule of thumb in watermarking is to spread the watermark to gain robustness against noise addition. This conflicts with the design of a domain invariant to geometric transformation.

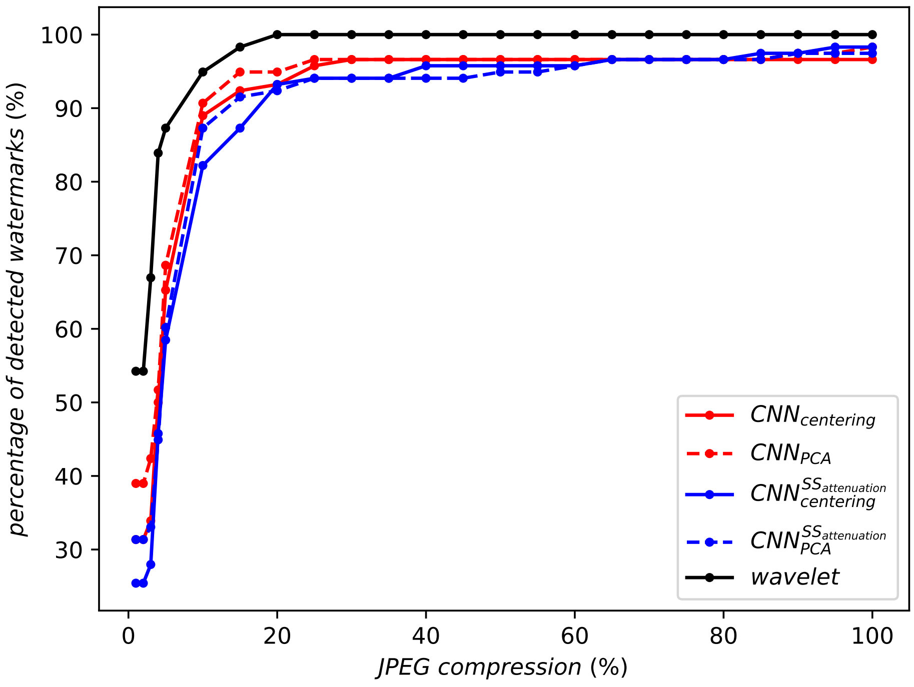

**Figure 7.** Robustness against JPEG compression.

## 5.5. Training Against Attacks

A new classification network is now trained from scratch. Its architecture is the same as the pretrained one used in the first part of our experiments, that is,VGG19. The ILSVRC2012 dataset that consists of 1.2 million images is augmented with attacked versions by the training pipeline. This pipeline introduces attacks of increasing strength. The network is trained for 1000 epochs which takes 3 weeks on an Nvidia GeForce GTX 1060 GPU with an Adam optimizer, a learning rate of 0.001, and a learning rate decay on plateau. The progression of the attacks is scheduled as follows:

- Initially, no attacks. Images come from the ILSVRC2012 dataset.
- After 5 epochs, horizontal flips and scaling within 90% and 110% are introduced.
- After 20 epochs, vertical flips and rotation of up to 3° (both clockwise and anticlockwise) are introduced. Scaling is extended to the range from 70% to 130%.
- After 35 epochs, rotation is extended up to ±5°.
- After 50 epochs, horizontal and vertical stretching up to 10% are introduced. Rotation is extended to ±10°, scaling to the ranges from 50% to 150%.
- From epoch 150 on, attacks are performed with the following parameters: rotation up to ±45°, scaling from 10% to 190%, horizontal and vertical stretching up to 50% and with horizontal and vertical flipping.

Training was performed in a standard classification setup with non-watermarked ImageNet images and is not related to the watermarking evaluation that follows.

Figure 8 illustrates the robustness to rotation of all the convolutional and pooling layers of the two VGG19 networks—the pretrained one (in black) and the one trained from scratch with progressive attacks (in red). Although there is a bit of overlapping for few layers, a clear separation is globally visible, indicating the better performances of the network trained with attacks. The best robustness is given by the pool4 layer of the network trained with attacks.

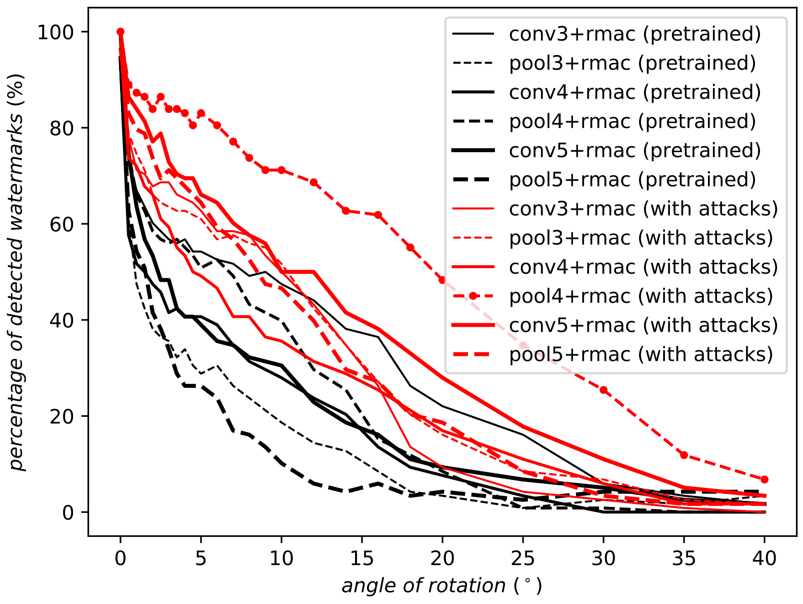

**Figure 8.** Robustness against rotation of different layers of the same architecture in a pretrained setup (black) and trained on a series of progressively harder attacks (red). Notation *conv*[*i*] indicates the output of the $i^{th}$ convolutional layer, *pool*[*i*] the output of the $i^{th}$ pooling layer of the VGG19 architecture, and $+rmac$ indicates that the output is passed through an RMAC aggregation layer (10).

## 5.6. Robustness with Aggregation

Aggregation methods improve the performance of convolutional neural networks in classification and content based image retrieval tasks. Is this still the case for watermarking?

When evaluating against rotation, WELDON and RMAC aggregations improve the robustness compared to the feature extraction without aggregation. Figure 9 moreover shows that RMAC clearly outperforms WELDON for large angles. Its robustness is more stable, smoothly degrading as the rotation angle increases.

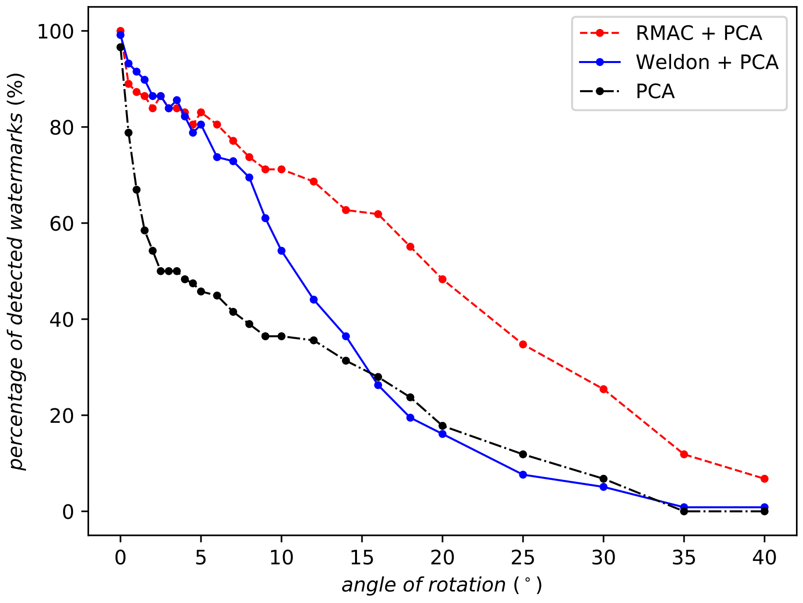

**Figure 9.** Robustness against rotation of the same network instance with different aggregation methods or no aggregation at all.

As for cropping, RMAC consistently outperforms WELDON and no aggregation at all as shows in Figure 10. For small cropping factors, the robustness of WELDON collapses more quickly than using no aggregation at all.

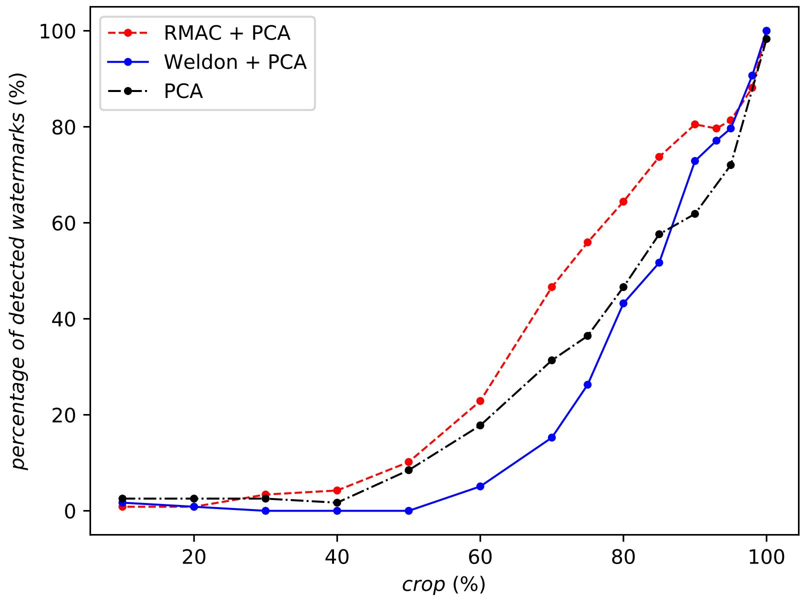

**Figure 10.** Robustness against cropping of the same network instance with different aggregation methods or no aggregation at all.

However, RMAC is not always the best aggregation. It is less robust than WELDON against JPEG compression (by a small amount—see Figure 11) and against a gamma correction (see Figure 12). Indeed, no aggregation at all seems to perform best against a gamma correction.

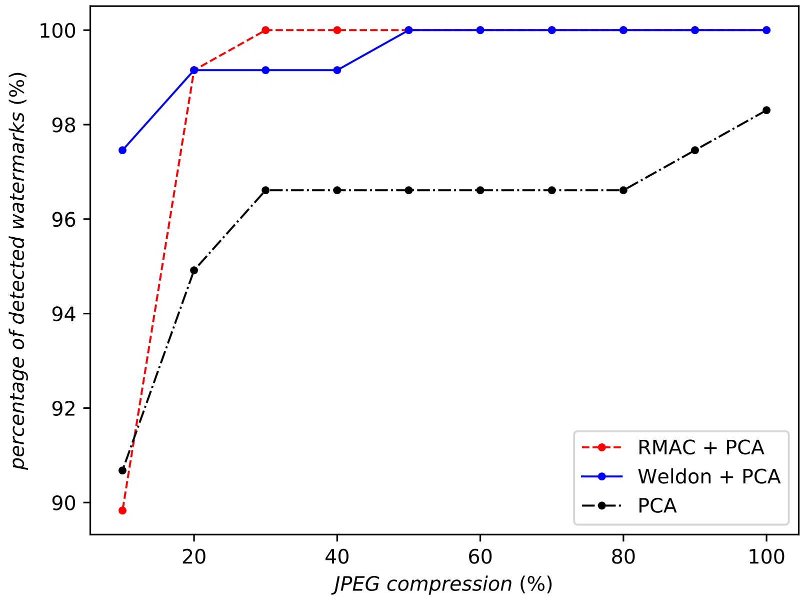

**Figure 11.** Robustness against JPEG compression of the same network instance with different aggregation methods or no aggregation at all.

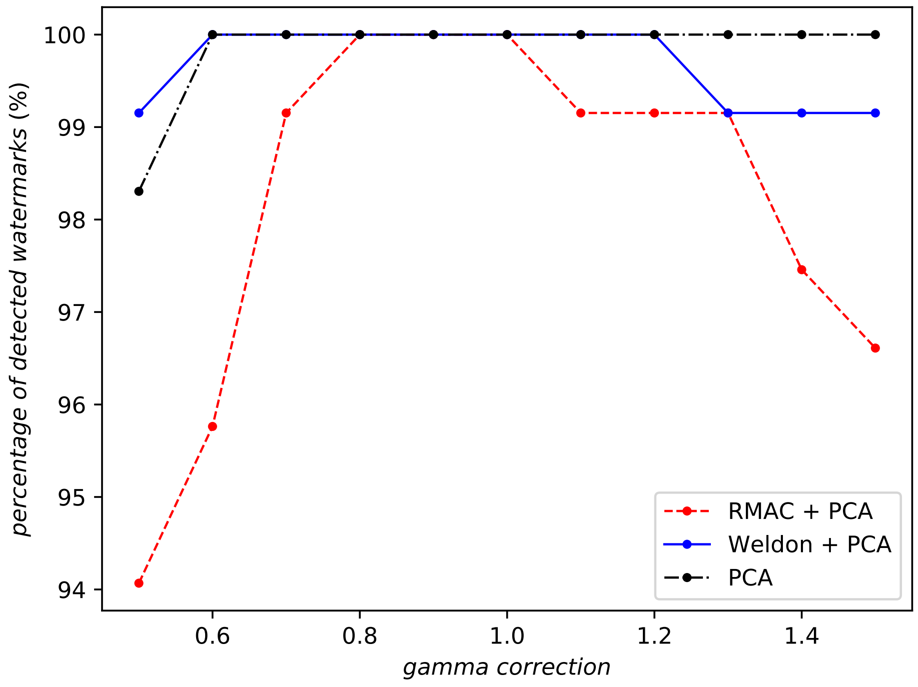

**Figure 12.** Robustness against gamma correction of the same network instance with different aggregation methods or no aggregation at all.

Note that Figures 9–12 are not using the same scale on the $y$-axis. Overall, RMAC is the obvious choice especially against geometric attacks. Its relative weakness against valuemetric attacks is certainly due to the different $\ell_2$ normalizations. Appendix A shows some watermarked images with this option.

## 6. Conclusions

This work shows that watermarking features extracted from a classification network is feasible and relevant. Feasible because the framework of adversarial sample provides a way to create watermarked image of good quality. Relevant because the DL network provides a transformation which strikes a good trade-off between invariance to geometric attacks and robustness to valuemetric attacks.

This robustness is further improved thanks to mechanisms borrowed from other Computer Vision task like RMAC in image search. The training dataset plays also a big role. Augmentation with specific image transformations increases the invariance of the network although the training is driven by the classification performance and oblivious to watermarking.

From the application point of view, this study opens the door to a single image descriptor good for both image search and watermarking. In copy detection and copyright infringement applications, image search alone yields many false positives. A watermark detection would drastically decrease the number of false recognition cases.

The proposed approach has some drawbacks. First, embedding is roughly 20 times slower than detection—it is iterative and each iteration computes a gradient by back-propagation, whose complexity is the double of one inference. Reference [14] does not have this shortcoming because a generative network is in charge of the embedding. On the other hand, this generative network is not secret-keyed. Second, watermarking is not purely invariant to geometric transformation. This is fine in some applications where watermarked images face mild geometric attacks. For other applications, it will need a registration mechanism, yet that mechanism does not need to be perfectly accurate.

**Author Contributions:** Conceptualization, V.V., V.C. and T.F.; methodology, T.F.; software, V.V.; validation, V.V. and V.C.; formal analysis, T.F.; investigation, V.V.; resources, V.V. and V.C.; writing–original draft preparation, V.V., V.C., and T.F.; writing–review and editing, T.F.; visualization, V.V.; supervision, T.F.; project administration, V.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Appendix A

This appendix shows some samples taken from the CLIP image dataset [29] (validation dataset 'professional' subset) watermarked with the following parameters: The network is the retrained model (Section 4.4) using the last VGG19 pooling layer (namely block pool4), followed by the aggregation RMAC (10) and a PCA reduction to $n = 256$ dimensions and normalization (Section 4.1). The target $P_{\mathsf{fa}}$ is $10^{-8}$, thus giving an angle $\theta = 69.7°$ in (21). The perceptual model based on SSIM attenuation is used (Section 4.3). The iterations use the step $\eta = 1.0$ and the stopping criterion was PSNR $< 43$ dB in Algorithm 1.

The figures display the original image on the left, its watermarked version on the right hand side, the PSNR in dB between the two, and the $p$-value measured when querying the watermarked version (see Section 4.2). Indeed, a low $p$-value reveals that the embedding succeeded in creating a watermarked image deep inside the detection region. Thus, for a given distortion budget, the lower this $p$-value, the easier to be watermarked is the image.

The images are grouped in three categories (highly—Figure A1, moderately—Figure A2, and almost not—Figure A3 textured images). The watermark is invisible except when the image is very flat like *davide-ragusa-716* in Figure A3. This image deserves a more advanced Human Visual System model than SSIM.

Surprisingly, moderately textured images are easier to be watermarked as their $p$-values suggest. We propose the following explanation. The embedding is very constrained on flat images because the perceptual model restricts the space (i.e., the number of pixels) where to hide the watermark signal. This concentrates all the distortion budget on few pixels and creates visible artefacts. For highly textured images, the inverse extraction function $\mathsf{Tr}^{-1}(\cdot)$ struggles to deviate the extracted feature strongly reflecting the textures present in the image within the low distortion budget. This is a pity because highly textured images have great masking capabilities. It means that they can bear large embedding distortions without perceptual degradation. A more advanced Human Visual System model adapts the distortion budget depending on the masking capabilities of the host image (i.e., the 'amount' of textures). This increases the perceptual quality of almost not textured images and at the same time it increases the robustness for highly textured images. We observe this effect when simply replacing the stopping condition based on the PSNR by SSIM $> 0.998$ in Algorithm 1.

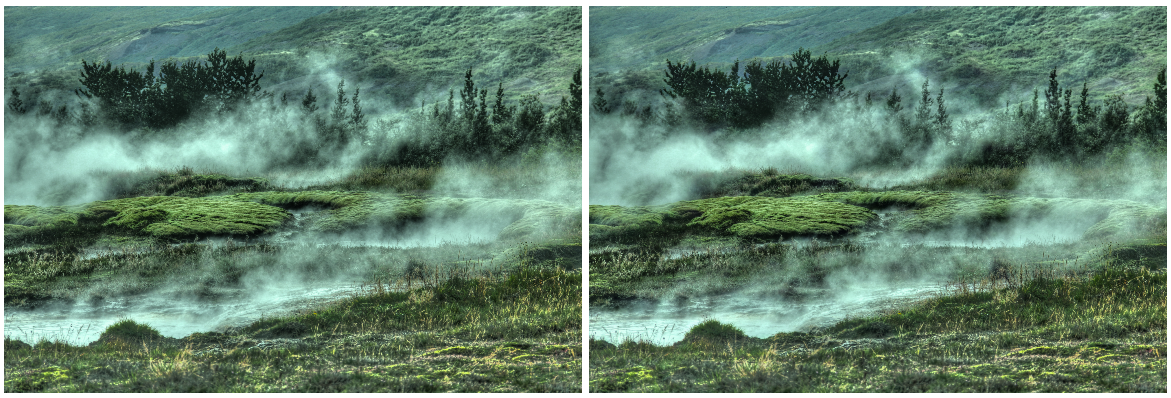

Image *gian-reto-tarnutzer-45212*. PSNR = 43.0 dB, SSIM = 0.998186, $P_{\mathsf{fa}} = 5.7 * 10^{-9}$.

**Figure A1.** *Cont.*

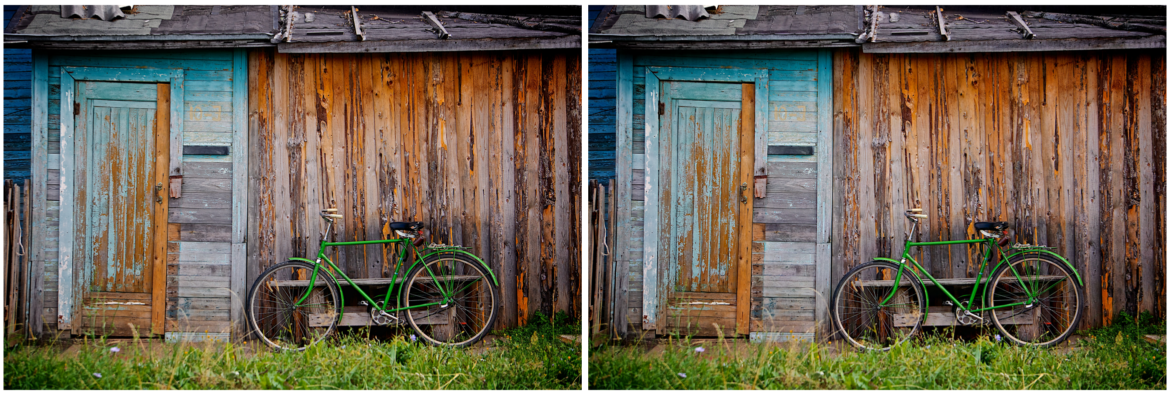

Image *alexander-shustov-73*. PSNR = 42.7 dB, SSIM = 0.998646, $P_{\mathsf{fa}} = 6.3 * 10^{-8}$.

**Figure A1.** Highly textured samples. (**Left**) Original image, (**Right**) Watermarked image.

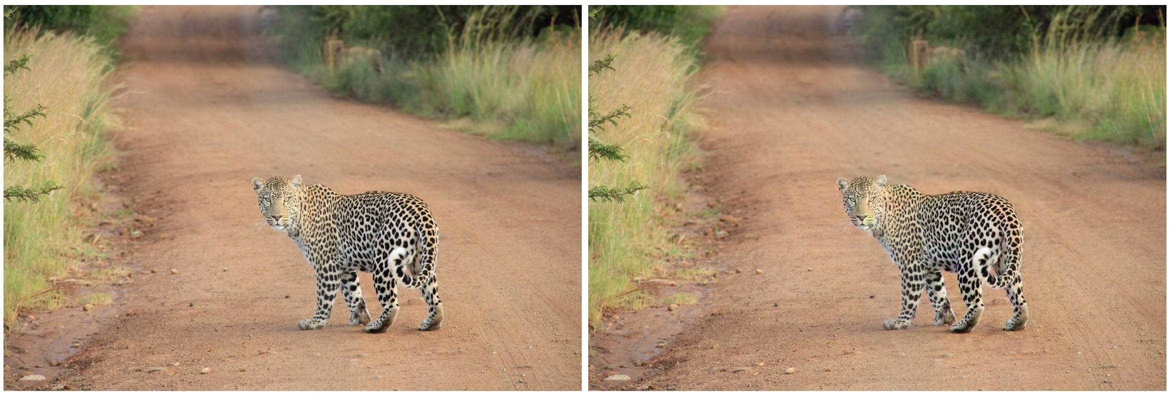

Image *martyn-seddon-220*. PSNR = 42.0 dB, SSIM = 0.995405, $P_{\mathsf{fa}} = 3.3 * 10^{-28}$.

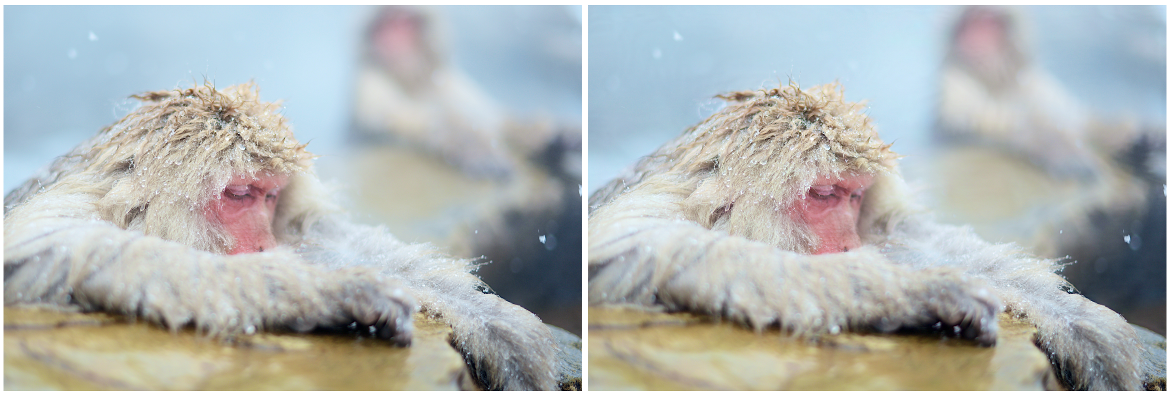

Image *nomao-saeki-33553*. PSNR = 42.7 dB, SSIM = 0.996335, $P_{\mathsf{fa}} = 5.7 * 10^{-15}$.

**Figure A2.** Moderately textured samples. (**Left**) Original image, (**Right**) Watermarked image.

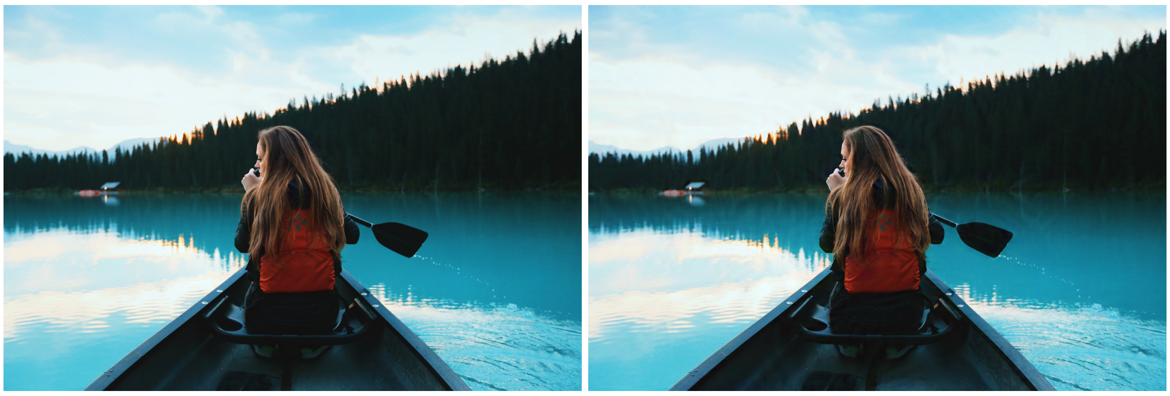

Image *roberto-nickson-48063*. PSNR = 43.0 dB, SSIM = 0.995713, $P_{\text{fa}} = 2.2 * 10^{-11}$.

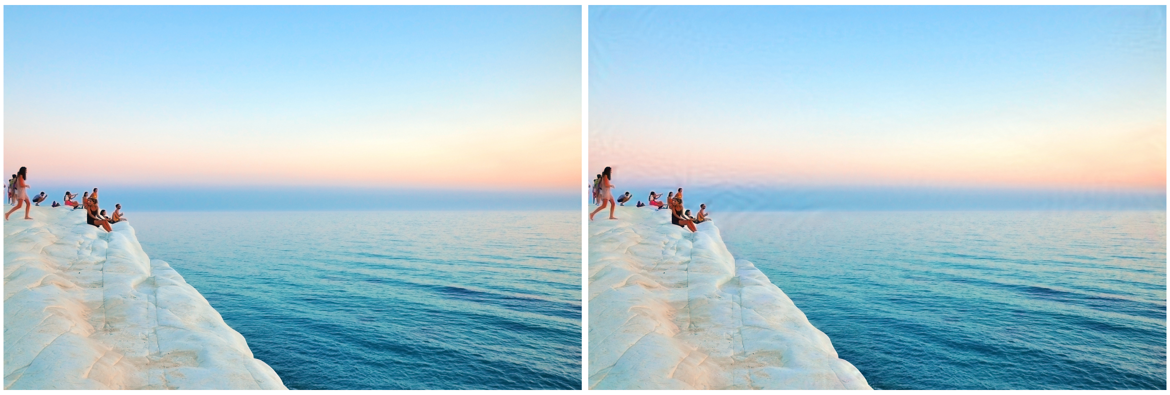

Image *davide-ragusa-716*. PSNR = 42.7 dB, SSIM = 0.995330, $P_{\text{fa}} = 6.3 * 10^{-11}$.

**Figure A3.** Almost not textured samples. (**Left**) Original image, (**Right**) Watermarked image.

## References

1. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **2012**, *60*. [CrossRef]
2. Siméoni, O.; Iscen, A.; Tolias, G.; Avrithis, Y.; Chum, O. Unsupervised object discovery for instance recognition. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV 2018), Lake Tahoe, NV, USA, 12–15 March 2018 .
3. Wu, J.; Yu, Y.; Huang, C.; Yu, K. Deep multiple instance learning for image classification and auto-annotation. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3460–3469.
4. Iscen, A.; Tolias, G.; Avrithis, Y.; Furon, T.; Chum, O. Panorama to Panorama Matching for Location Recognition. In Proceedings of the ACM International Conference on Multimedia Retrieval (ICMR 2017), Bucharest, Romania, 6–9 June 2017; ACM: New York, NY, USA, 2017.
5. Radenović, F.; Tolias, G.; Chum, O. CNN Image Retrieval Learns from BoW: Unsupervised Fine-Tuning with Hard Examples. In *Computer Vision—ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; pp. 3–20.
6. Gordo, A.; Almazán, J.; Revaud, J.; Larlus, D. Deep image retrieval: Learning global representations for image search. In *Computer Vision – ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; pp. 241–257.
7. Kanbak, C.; Moosavi-Dezfooli, S.M.; Frossard, P. Geometric robustness of deep networks: Analysis and improvement. *arXiv* **2017**, arXiv:1711.09115 .

8. Nagai, Y.; Uchida, Y.; Sakazawa, S.; Satoh, S. Digital watermarking for deep neural networks. *Int. J. Multimed. Inf. Retr.* **2018**, *7*, 3–16. doi:10.1007/s13735-018-0147-1. [CrossRef]
9. Darvish Rouhani, B.; Chen, H.; Koushanfar, F. DeepSigns: A Generic Watermarking Framework for IP Protection of Deep Learning Models. *arXiv* **2018**, arXiv:1804.00750.
10. Yu, P.T.; Tsai, H.H.; Lin, J.S. Digital watermarking based on neural networks for color images. *Signal Process.* **2001**, *81*, 663–671. doi:10.1016/S0165-1684(00)00239-5. [CrossRef]
11. Khan, A.; Tahir, S.F.; Majid, A.; Choi, T.S. Machine learning based adaptive watermark decoding in view of anticipated attack. *Pattern Recognit.* **2008**, *41*, 2594–2610. doi:10.1016/j.patcog.2008.01.007. [CrossRef]
12. Stolarek, J.; Lipiński, P. Improving digital watermarking fidelity using fast neural network for adaptive wavelet synthesis. *J. Appl. Comput. Sci.* **2010**, *18*, 61–74.
13. Mun, S.; Nam, S.; Jang, H.; Kim, D.; Lee, H. A Robust Blind Watermarking Using Convolutional Neural Network. *arXiv* **2017**, arXiv:1704.03248.
14. Zhu, J.; Kaplan, R.; Johnson, J.; Li, F.-F. HiDDeN: Hiding Data With Deep Networks. In *Computer Vision–ECCV 2018*; Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y., Eds.; Springer: Cham, Switzerland, 2018; pp. 682–697.
15. Quiring, E.; Arp, D.; Rieck, K. Forgotten siblings: Unifying attacks on machine learning and digital watermarking. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy, London, UK, 24–26 April 2018.
16. Vukotić, V.; Chappelier, V.; Furon, T. Are Deep Neural Networks good for blind image watermarking? In Proceedings of the 2018 IEEE International Workshop on Information Forensics and Security (WIFS), Hong Kong, China, 11–13 December 2018; pp. 1–7.
17. Merhav, N.; Sabbag, E. Optimal watermark embedding and detection strategies under limited detection resources. *IEEE Trans. Inf. Theory* **2008**, *54*, 255–274. [CrossRef]
18. Furon, T.; Bas, P. Broken Arrows. *EURASIP J. Inf. Secur.* **2008**, *2008*, 597040. doi:10.1155/2008/597040. [CrossRef]
19. Comesana, P.; Merhav, N.; Barni, M. Asymptotically optimum universal watermark embedding and detection in the high-SNR regime. *IEEE Trans. Inf. Theory* **2010**, *56*, 2804–2815. [CrossRef]
20. Cox, I.; Miller, M.; Bloom, J. *Digital Watermarking*; Morgan Kaufmann: Burlington, MA, USA, 2002.
21. Tolias, G.; Sicre, R.; Jégou, H. Particular object retrieval with integral max-pooling of CNN activations. *arXiv* **2015**, arXiv:1511.05879 .
22. Durand, T.; Thome, N.; Cord, M. Weldon: Weakly supervised learning of deep convolutional neural networks. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 4743–4752.
23. Moosavi-Dezfooli, S.M.; Fawzi, A.; Frossard, P. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
24. Carlini, N.; Wagner, D. Towards Evaluating the Robustness of Neural Networks. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 39–57. doi:10.1109/SP.2017.49. [CrossRef]
25. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples. *arXiv* **2014**, arXiv:1412.6572.
26. Kurakin, A.; Goodfellow, I.; Bengio, S. Adversarial examples in the physical world. *arXiv* **2016**, arXiv:1607.02533.
27. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. doi:10.1007/s11263-015-0816-y. [CrossRef]
28. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2015**, arXiv:1409.1556.

29. Challenge on Learned Image Compression. Available online: http://www.compression.cc/challenge/ (accessed on 10 June 2018).
30. Alakuijala, J.; Obryk, R.; Stoliarchuk, O.; Szabadka, Z.; Vandevenne, L.; Wassenberg, J. Guetzli: Perceptually Guided JPEG Encoder. *arXiv* **2017**, arXiv:1703.04421.