FlexMark: Adaptive Watermarking Method for Images

Mohammad Amin Arab
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada

Ali Ghorbanpour
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada

Mohamed Hefeeda
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada

ABSTRACT

Most current watermarking methods offer low and fixed capacity, which means they can only embed small-size watermarks into images. Additionally, they are typically robust to only a small subset of the known image transformations (aka distortions) that occur during the processing, transmission, and storage of images. These shortcomings limit their adoption in many practical multimedia applications. We propose FlexMark, a robust and adaptive watermarking method for images, which achieves a better capacity-robustness trade-off than current methods and can easily be used for different applications. FlexMark categorizes and models the fundamental aspects of various image transformations, enabling it to achieve high accuracy in the presence of many practical transformations. FlexMark introduces new ideas to further improve the performance, including double-embedding of the input message, employing self-attention layers to identify the most suitable regions in the image to embed the watermark bits, and utilization of a discriminator to improve the visual quality of watermarked images. In addition, FlexMark offers a parameter, α, to enable users to control the trade-off between robustness and capacity to meet the requirements of different applications. We implement FlexMark and assess its performance using datasets commonly used in this domain. Our results show that FlexMark is robust against a wide range of image transformations, including ones that were never seen during its training, which shows its generality and practicality. Our results also show that FlexMark substantially outperforms the closest methods in the literature in terms of capacity and robustness.

CCS CONCEPTS

• **Applied computing** → *Computer forensics*; • **Information systems** → *Multimedia information systems*.

KEYWORDS

Image Watermarking, Steganography

ACM Reference Format:
Mohammad Amin Arab, Ali Ghorbanpour, and Mohamed Hefeeda. 2024. FlexMark: Adaptive Watermarking Method for Images. In *ACM Multimedia Systems Conference 2024 (MMSys '24), April 15–18, 2024, Bari, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3625468.3647611

MMSys '24, April 15–18, 2024, Bari, Italy

ACM ISBN 979-8-4007-0412-3/24/04
https://doi.org/10.1145/3625468.3647611

1 INTRODUCTION

Identifying authentic content and detecting copyright infringements have become major concerns in recent years, as advancements in machine learning have made creating synthetic/fake content a relatively easy task. One way to protect the integrity and ownership of content, e.g., an image, is to embed a special message into it, called a *watermark*. In many applications, it is required that the embedded watermark should not change the visual appearance of the original image, i.e., it should be *imperceptible* to humans. The maximum size of the watermark that is imperceptible is known as the *capacity* of the watermarking system.

In addition, multimedia objects such as images go through many *transformations* (aka distortions) during the storage and transmission processes. Some of these transformations are legitimate and part of normal processing, such as compression. Other transformations are intentionally (or maliciously) introduced to evade the detection of illegally copied or manipulated images. Examples of such transformations include cropping, blurring, and changing the color space of images. To be of practical use, a watermarking system should be *robust* to the various transformations that might occur on images. Traditional watermarking systems, e.g., [4, 14], embed the watermark into the least significant bits of images to achieve imperceptibility. This, however, does not provide robustness against transformations.

Recent machine-learning-based watermarking methods, e.g., [21, 23, 30], embed and spread the watermark across the bits of an image while meeting a given imperceptibility and visual quality objective. However, while these methods are more robust to transformations, they offer limited capacity. Furthermore, current methods are typically designed and trained on a given watermark size and a fixed set of transformations. Thus, they cannot easily be adapted to multimedia applications with different authentication and data-hiding requirements, which limits their applicability in practice. For example, some applications, such as copyright protection [25], broadcast monitoring [12], and content authentication [2], require maximum robustness to transformations, but they do not need high capacity. Other applications, e.g., privacy protection in tele-medicine [22], require hiding as much data as possible within an image while restricting the allowed transformations on the content and thus do not need much robustness to transformations. A third class of applications, e.g., stealthy communications [11], lies between the aforementioned two extremes and requires a balance between robustness and capacity. We illustrate the spectrum of various requirements in Figure 1. We note that hiding significant amounts of information within images is sometimes referred to as *steganography*.

Current methods in the literature either trade off capacity for robustness or the other way around. For example, the methods

Figure 1: Examples of multimedia applications with different data hiding requirements.

in [8, 18] offer high capacity, but they do not tolerate any transformations. Conversely, the methods in [7, 17, 39] are robust to several types of transformations, but they can only offer limited capacity. Further, none of the existing methods can be customized for different applications.

In this paper, we propose FlexMark, which is a watermarking method that offers both high capacity and strong robustness to all practical transformations. FlexMark is also *adaptable* and can easily be customized for different multimedia applications. Achieving robustness, high capacity, and adaptability while maintaining image quality is a challenging research problem. Robustness, for instance, requires handling various transformations. However, there are numerous types of transformations, and each comes with different parameters and configurations. For example, image compression can be done using various codecs, e.g., JPEG and WebP, and each codec can produce different levels of quality. Similarly, the degree of blurriness in Gaussian blur and the kernel size in image filtering can have many different values. Thus, it is hard to enumerate all possible transformations, let alone create datasets to train a machine-learning model to handle them. To address this problem, we divide all transformations into four main groups and model the *fundamental* aspects of each group. Then, we train our model only on a few *samples with randomized parameters* from each group.

To increase the capacity of the watermarking system without damaging the visual quality of images, we introduce multiple ideas in FlexMark, including: (i) a message expansion method that allows for hiding different amounts of information, (ii) embedding the expanded information at two different stages of the model to capture low- and high-level features, and (iii) using a self-attention module to identify the best locations in the image to embed the information and design multiple loss functions. To achieve adaptability, we design FlexMark with a configurable parameter, α, which controls the trade-off between the capacity and robustness. This allows FlexMark to meet the requirements of different applications, *without* changing or re-training it.

We implement FlexMark and assess its performance on standard datasets commonly used in this domain. Our results show that FlexMark provides consistently high accuracy and robustness to all practical transformations, including the ones that it did not see in the training dataset. Specifically, we train FlexMark on only five transformations and show its accuracy and robustness for sixteen different transformations. In addition, we compare FlexMark against the state-of-the-art methods [1, 7, 39], and our results show that it outperforms them in terms of capacity and robustness.

The main contributions of this paper are as follows:

- We propose a robust, adaptable, and high-capacity watermarking method that can be used with different applications without re-training.
- We categorize all practical image transformations and model their fundamental aspects, which significantly improves the robustness of watermarking systems without requiring large datasets for training.
- We present multiple ideas to improve the capacity of watermarking systems without damaging the visual quality of images, such as expanding and double-embedding the input message as well as employing a self-attention module to identify the most suitable regions in the image to embed the watermark bits.
- We conduct rigorous evaluations with standard datasets to demonstrate the accuracy and robustness of FlexMark. Our results also show that FlexMark outperforms the state-of-the-art methods in the literature.
- We conduct an ablation study to analyze the performance contribution of each component of FlexMark.

2 BACKGROUND AND RELATED WORK

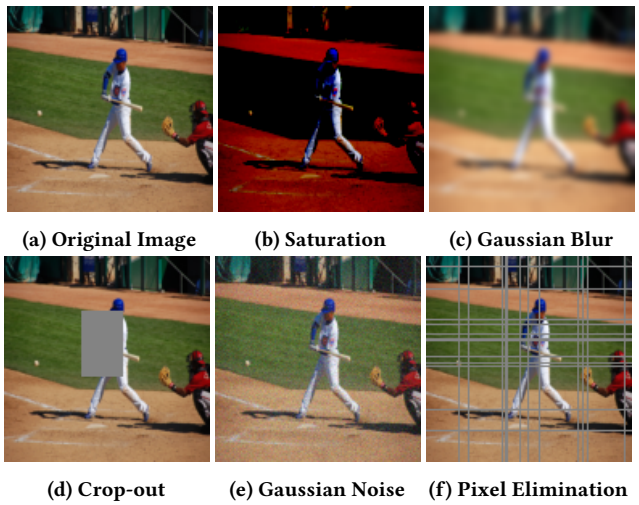
Terminologies and Symbols. Traditionally, watermarking and steganography are considered two different but related research topics. However, the requirements of modern multimedia applications and the advent of deep learning have blurred the boundaries between them. We use the term *data hiding* to abstractly refer to all variations of watermarking and steganography methods.

In a data hiding system, a *message* M is embedded (or hidden) into a *cover image* I_{co} by an encoder, and the result is referred to as the *encoded image* I_{en}. Images I_{co} and I_{en} should be visually indistinguishable by humans. This is referred to as the imperceptibility requirement of data-hiding systems.

The encoded image I_{en} may then be subjected to various *transformations*, such as compression, cropping, color changes, and noise additions during the storage and transmission pipeline. Transformations are sometimes referred to as distortions or attacks in the literature. Sample transformations are shown in Figure 2. We denote the encoded image after transformations by I'_{en}, which is then transferred to receivers. Receivers use a decoder to retrieve the embedded message M from the noised image I'_{en}.

The *capacity*, in bits per pixel (bpp), of a data hiding system is defined as the maximum size of the message M that can be embedded in the pixels of the cover image I_{co}. We call a data hiding system *robust* against a specific transformation if the retrieved message by the decoder M' is similar to the originally embedded message M by the encoder, given that the encoded image was subjected to that specific transformation. The similarity between M and M' is measured by the number of matching bits in them. Error-correcting codes are usually used in practice to tolerate a few bit errors in M'.

Watermarking. Image watermarking is the process of embedding a message (watermark) into an image. It has many applications, including content authentication and copyright protection. Content authentication is used to verify the source and ownership of an image, as well as to ensure that the image has not been tampered with. This is especially important for legal and forensic purposes



(a) Original Image (b) Saturation (c) Gaussian Blur

(d) Crop-out (e) Gaussian Noise (f) Pixel Elimination

Figure 2: Sample image transformations.

as well as for protecting the image owner's reputation. Copyright protection, on the other hand, is used by content creators to track the use of their images. This helps prevent unauthorized usage of copyrighted images and ensures owners receive proper credit for their work [6, 29].

Most of the recent watermarking methods use deep neural networks. For example, HiDDeN [39] embeds binary messages of 30 bits into an image of size 128×128 pixels, which is a capacity of around 0.002 bpp. Similarly, the method in [19] has a fixed capacity of 0.002 bpp. ReDMark [1] uses two deep residual neural networks for embedding and extracting watermarks, which improves the capacity and robustness of their system to different types of transformations, including cropping and JPEG compression. FIN [7] employs invertible neural networks instead of the conventional encoder/decoder architectures, which enables it to offer a larger capacity.

Steganography. Image steganography is a method for concealing information within a cover image. It is used in applications such as stealthy communications [20] and privacy protection of medical records [13]. In stealthy communications, an image is used as a *covert* channel for discretely exchanging messages. Since images can be easily shared over open networks such as the Internet without attracting suspicion, steganography offers an attractive method for secretly exchanging information [3]. As an example, a report by the Federal Bureau of Investigation (FBI) in the United States described how the Russian Foreign Intelligence Services (SVR) employed image steganography to transmit messages to some of their agents by embedding these messages into images and posting them on open websites [26].

Similar to watermarking, most recent steganography methods use deep learning. For example, several methods, e.g., [5, 27, 34, 35], use an encoder-decoder structure that is based on the U-Net model. Other methods, e.g., [9, 28, 32, 33], use GANs (Generative Adversarial Networks), in which a generator/discriminator duo help each other to generate the results. Invertible neural networks have also been used in image steganography [10, 16, 18, 36]. In addition, multiple works have proposed ideas to improve the robustness of steganography methods. For example, Tao et al. [31] design a coefficient adjustment scheme that is robust to JPEG compression. Zhang et al. [38] utilize a burst error model to improve robustness. Qian et al. [24] propose a steganography method based on texture synthesis that can provide robustness against JPEG compression, which is based on hiding the message bits inside some selected tiles of each image.

State of the Art Methods Considered for Comparisons. ReDMark [1] and FIN[7] represent the current state of the art. These methods have been shown to outperform others in the literature. We consider ReDMark because it is the closest work that provides variable capacity. We compare against FIN since it is the most recent method we are aware of and is based on invertible neural networks that offer high capacity.

3 PROPOSED SOLUTION

In this section, we first summarize the design goals of FlexMark and how it achieves them. Then, we present an overview of its operation and main components. This is followed by the details of each component.

3.1 Design Goals

We summarize the design goals of FlexMark and our approaches to achieve them in the following:

- *Adaptable to different applications.* As described in §2, multimedia applications have diverse data hiding requirements. Unlike current methods in the literature, FlexMark offers a knob, α, through which it can control the trade-off between the capacity and robustness, and hence it can easily be customized to different applications.
- *Maintain visual quality.* Embedding a message into an image results in changing some of its pixels, which could introduce distortions in the encoded image or even visually change it significantly from the input cover image. To achieve high visual quality, we design FlexMark as a generative model with a discriminator that ensures the produced encoded image is visually as close as possible to the input cover image.
- *Robust to various transformations without compromising capacity.* It is crucial for a data hiding system to be robust to transformations that might occur on the encoded image during transmission and storage. Prior methods sacrificed substantial capacity to partially achieve this robustness. In contrast, FlexMark introduces new ideas, e.g., double message embedding and multiple custom loss functions, to achieve a better robustness-capacity trade-off.
- *Expandable to new transformations.* Transformations on images may evolve with time. For example, newer JPEG 2000 and WebP compression algorithms produce different outputs than JPEG. Thus, a data hiding system designed to be robust for JPEG may not function properly for JPEG 2000 and WebP. Therefore, unlike prior works that may not easily support new transformations, FlexMark categorizes and models the fundamental effects of all transformations, which allows it to be robust against transformations not seen during training.

3.2 Overview of FlexMark

Figure 3 presents an overview of FlexMark, which has multiple deep neural network models colored green. All components are trained together, end-to-end. FlexMark presents multiple new ideas beyond prior works, e.g., [1, 7, 39], including efficient modeling of diverse image transformations, double-embedding of the input message to increase robustness, a flexible message expansion method that allows different capacities, and new designs of the encoder and decoder neural network models. After training, FlexMark functions as follows. A content creator embeds a message M into a cover image I_{co} using the Encoder, which is a deep neural network model (described in §3.3). The Encoder operation is controlled by the parameter β. Before embedding, the input message $M \in \{0,1\}^D$ is expanded to be the same dimensions as the cover image, which are $H \times W \times 3$. The expanded message is referred to as I_M. This expansion process is controlled by the parameter α. The two parameters α and β control the trade-off between the capacity of the data hiding system, the level of robustness to transformations, and the visual quality of the produced cover images. When a user receives an image with an embedded message, it uses the Decoder to retrieve this message.

The encoded image I_{en} will likely be subjected to various transformations before it arrives at the receiver. This means that the Decoder should be able to recover the embedded message even in the presence of transformations. To achieve this, we model the effects of different transformations. There are many possible transformations, and each can come with different strengths and parameters. For example, the quality level used in compression and the kernel size of the image blurring filter can each have many different values. This can lead to numerous combinations of transformations, which are not possible to enumerate and train the model on. To address this problem, we categorize all transformations into four main classes. We then model only the essential characteristics of these classes and train the model on them. This enables our model to generalize to different transformations, including the ones that were never seen during training.

In addition, some of the transformations, e.g., adding Gaussian noise to the encoded image, are differentiable and hence can easily be added to the training loop of the whole model, while others, e.g., JPEG compression, are not differentiable. We design and *pre-train* a neural network model for non-differentiable transformations. §3.4 presents the details of modeling the impact of transformations.

Embedding a message M into the cover image could introduce visual artifacts. Further, transformations of the encoded image could change bits of the embedded message. We design multiple loss functions to maximize the visual quality of the encoded image and the accuracy of the retrieved message M', as detailed in §3.5.

3.3 Design of the Encoder and Decoder

Design of the Encoder. The Encoder is a deep neural network model that embeds the message M into the cover image I_{co}. M is first expanded to the same dimensions of I_{co}, which we then refer to as I_M. As described later, this message expansion is controlled by α. Inspired by the U-NET [27] structure, we design the Encoder to have a contracting path and an expansive path. During the contracting path, low-level features are extracted until the feature maps are distilled into a small cube of data. Then, the high-level features are extracted during the expansive path.

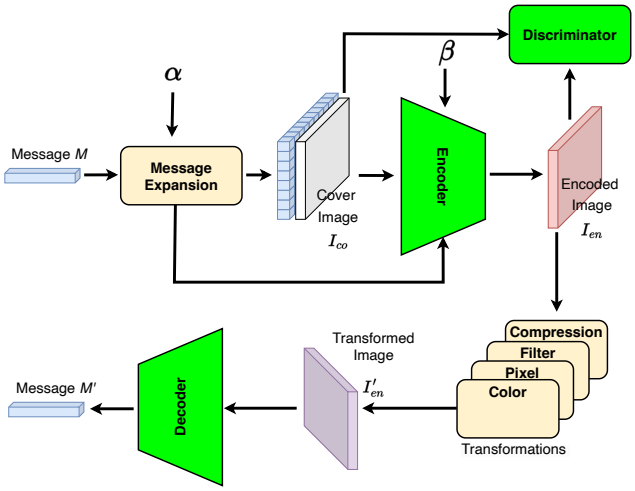


Figure 3: Overview of FlexMark.

The design of the Encoder allows it to provide robustness to various transformations. This is because some transformations, e.g., additive noise, affect high-level features while others, e.g., blurring, distort low-level features. To further improve the robustness of FlexMark to transformations that could affect the high- and/or low-level features, the Encoder embeds the input message twice at different locations.

The input to the Encoder is the concatenation of I_M and I_{co}. I_{co} is concatenated with I_M along the channel dimension to create a six-channel 3-D cube of shape $H \times W \times 6$. This cube goes through two layers of convolution, batch normalization, and ReLU. Then, the contraction starts. During each step of contraction, a skip connection is added to the corresponding layer in the expansion stage. Skip connections preserve the gradient signal by allowing it to bypass some layers, which improves the training efficiency of the network. During the expansive path, we add a weighted version of the expanded message to the feature maps to preserve the hidden message in high-level features of the generated image, which is given by:

$$I_{L-1} = I_{L-2} + \beta * I_M, \tag{1}$$

where L is the number of hidden layers in the encoder, and β is the factor controlling the trade-off between visual quality and robustness. Higher β values make I_M easier to recover by the Decoder, i.e., higher robustness, but they reduce the visual quality. Lower β values result in higher-quality images, but they make it harder for the Decoder to retrieve the embedded message. We conducted experiments to analyze the impact of β and determine the most suitable value to optimize the performance of FlexMark. Specifically, we trained FlexMark on different values of β between 0 and 0.15, and measured the visual quality (quantified by PSNR and SSIM of the encoded image) and the bit accuracy of the retrieved message. Our results showed that the bit accuracy did not improve for β values greater than 0.01, while the visual quality continued to drop as we increased β. Thus, we set $\beta = 0.01$ for all of our experiments

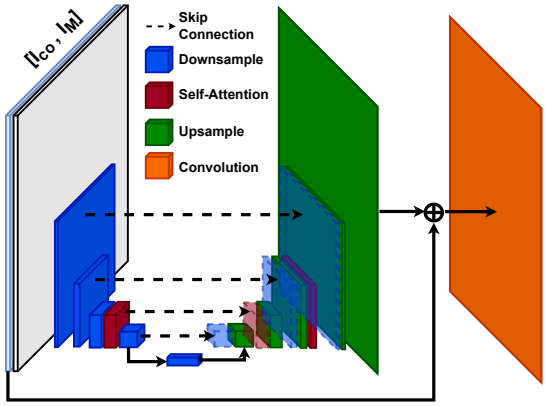


Figure 4: Design of the Encoder in FlexMark. Key design aspects include: (i) using self-attention to capture the dependencies across image regions and (ii) double-embedding the input message to improve the robustness to transformations.

α	Block Shape	Message Length
128	128×128	1
64√2	128×64 or 64×128	2
64	64×64	4
	...	...
1	1×1	16384

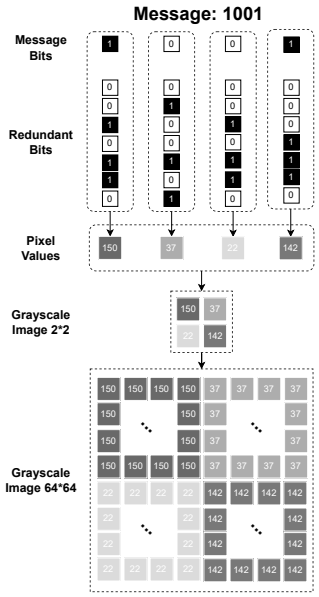


Figure 5: Message expansion in FlexMark. Various values of α and their corresponding block and message sizes are shown on the left. The illustration on the right is for a message size of 4 bits and block size of 64×64. Each bit of the message is padded with random bits to create an 8-bit pixel. Pixels are then duplicated to form 64×64 blocks, which are concatenated together to form the expanded message I_M.

in this paper. In addition, the Encoder uses the self-attention mechanism [37] to improve the performance of our design. Self-attention is complementary to the structure of the encoder. It models the multi-level, long-range dependencies that usually exist across the various image regions. Thus, it allows the network to explore areas further than the scope of common convolution layers to find the most suitable locations to embed the message, helping to achieve one of the main objectives of data-hiding systems, which is imperceptibility. The final activation layer of the encoder is a TanH function.

Message Expansion. We elaborate on the process of expanding the message M into a format suitable for embedding within the cover image I_{co}, characterized by dimensions $H \times W \times 3$. Our design aims to offer an adaptive balance between the robustness and capacity of the watermarking system, a balance regulated by the parameter α.

The message M is expanded into an image referred to as I_M, which has the same dimensions as I_{co}. Specifically, I_M is logically partitioned into blocks, with each block carrying a single bit of information from M. The block size is denoted by $w \times h$, where w and h are determined by the dimensions of I_{co} and α. We define α as the square root of the area of the blocks: $\alpha = \sqrt{w \times h}$, where w and h range from 0 to W and H, respectively. Notice that a larger α corresponds to larger blocks and, as a result, fewer number of blocks within I_{co}. Since only one bit is embedded in each block, the capacity is given by $\frac{W \times H}{\alpha^2}$ bits. Increasing the number of blocks increases the system's capacity, but it makes it harder for the decoder to extract the secret bit from a smaller area, especially in the presence of distortions. Thus, the α parameter plays a crucial role in controlling the trade-off between system robustness and capacity.

We illustrate the message expansion process in Figure 5 for a small message of 4 bits expanded into a cover image of size 128×128, which is divided into 4 blocks each of size 64×64. α is set to 64 in this case. The figure also shows a table of some possible values of α and their corresponding block sizes. As the figure shows, each bit of the input message is first padded with 7 random bits to form a gray-scale pixel. Then, each pixel value is duplicated to fill a block of size 64×64.

Robustness via Double Embedding. We note that, unlike most prior works in the literature, we expand the message to create a secret image before we concatenate it with the cover image. So, instead of combining the message with the cover image only in the hidden layers of the Encoder, we also concatenate the expanded secret message with the cover image before feeding it to the Encoder. This *double embedding* mechanism is one of the reasons for the robustness of FlexMark against a wide range of transformations. This is because double embedding inserts the message both in the deep and shallow layers of the network. Since most transformations affect one or a few features of the encoded image, FlexMark can retrieve the message even in the presence of several transformations.

Capacity-Robustness Adaptation. FlexMark uses α to control the trade-off between capacity and robustness. This flexibility allows FlexMark to meet the data-hiding requirements of different applications across many applications. For example, in scenarios such as covert communications or tele-medicine, visual quality and capacity are the most important factors. To meet these requirements, FlexMark uses large α. Smaller α corresponds to smaller block sizes, which means higher data-hiding capacities. However, smaller blocks make the decoder more susceptible to distortions and as a result, make the system less robust. On the contrary, for applications such as copyright protection or integrity verification, robustness is the most important aspect. Thus, FlexMark sets α

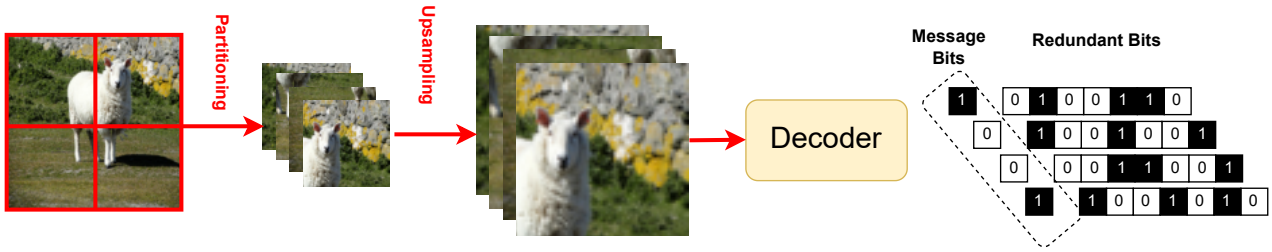


Figure 6: The operation of the Decoder in FlexMark. The illustration is for an image embedded with 4 bits of data (1001). First, the watermarked image is partitioned into blocks. Each partition is upsampled to the original image size. Then, the Decoder extracts one bit of information alongside the padding bits from each block.

to smaller values for higher robustness in such applications. Figure 1 illustrates a spectrum of applications that can be effectively supported by FlexMark.

Design of the Decoder. The Decoder's structure is similar to the first half of the encoder. Figure 6 shows the decoding phase of FlexMark. The purpose of the Decoder is to extract the secret message from each block of the encoded image that hides an 8-bit message chunk. Each block contains 1 bit of information and 7 bits of random padding, ensuring the independence of the system from the least significant values in the pixel as they are vulnerable to distortions. The Decoder output's layer consists of 8 nodes. The output is an 8-bit binary code, which has to equal the 8-bit of the corresponding one bit of the secret message and seven bits of padding. Since our system is adaptable to different message sizes, the Decoder needs to be able to handle different block sizes. We solve this issue by upsampling all block sizes to the original image size before feeding it into the Decoder. Interpolating smaller blocks to the largest size possible allows FlexMark to undergo training *just once*, accommodating different message sizes.

3.4 Modeling Image Transformations

Transformations are changes that occur on encoded images during storage or transmission. These transformations can be intentional or malicious to evade the detection mechanism in a watermarking system. In this case, they are referred to as attacks. Transformations can also occur because of normal processing, e.g., compression, or noise on the transmission channel, e.g., Gaussian noise. In the latter case, transformations are sometimes referred to as distortions. Example image transformations are shown in Figure 2.

We model the impact of transformations with the goal of efficiently and robustly covering the wide range of existing transformations in the literature as well as new ones that may be introduced in the future. To achieve this goal, we focus on modeling the *fundamental* aspects of transformations. That is, we model how various transformations could potentially affect an image. For example, some transformations affect all pixels in the image, e.g., Gaussian Blur, while others only alter a fraction of the pixels, e.g., pixel elimination. Furthermore, some transformations involve the frequency domain, e.g., compression, and others impact the color space of the image, e.g., changing brightness.

We propose dividing all image transformations into four categories: Compression, Image Filtering, Color Changes, and Local Pixel Changes. We describe each category in the following.

Compression. Compression is one of the most important transformations that occur in the transmission/storage pipeline. Most compression methods involve multiple steps, including color conversion, sub-sampling, block-processing, domain transformation, quantization, and length encoding. Some of these steps, e.g., quantization, are non-differentiable, which makes the whole compression transformation non-differentiable. Thus, it cannot be implemented in the neural network pipeline, as this would stop the gradient propagation.

To address this problem, we design a neural network model to *approximate* the effects of different compression methods and non-differentiable transformations in general. The model has 23 layers and is structured as a U-Net [27]. The contracting/expansive paths allow the model to learn subtle differences between compressed and non-compressed images.

We train the transformation approximation model using pairs of images, with and without the considered non-differentiable transformation(s). Once this model is trained, it is included in the pipeline of training FlexMark as shown in Figure 3.

The performance of the proposed transformation approximation model is summarized in Table 1, where we show the differences (in terms of L1 and L2 losses) between images compressed by actual WebP and JPEG2000 encoders and the same images processed by our model without any compression. These results are averaged over 5,000 images, and they show that the compressed images are very similar to the approximated ones.

	JPEG	WebP	JPEG 2000
L_1 Loss Value	0.007	0.011	0.006
L_2 Loss Value	0.00013	0.00025	0.00010

Table 1: Performance of the proposed neural network model to approximate non-differentiable transformations such as JPEG, WebP, and JPEG2000 compression. L_1 and L_2 are computed between each compressed and its approximated one.

Using the proposed neural network model, various non-differentiable transformations and compression methods can be handled, including the following:

- *JPEG: One of the most widely used compression.
- JPEG 2000: Uses the discrete wavelet transform instead of the discrete cosine transform in JPEG.
- WebP: Introduced by Google for web page images, and it makes multiple changes to the de-quantization, prediction, and length encoding steps of JPEG.
- Other non-differentiable transformations.

We note that transformations marked by * are the ones used in the training of our watermarking method.

Image Filtering. Another common group of transformations can be collectively modeled by image filtering. Generally, a filter sweeps the image and convolves it with a kernel. Image filtering usually affects all pixels in the image, and it can be implemented using differentiable functions. Examples that can be handled by this category include the following:

- *Gaussian Blur: Pixels close to the center of the kernel are given higher weights than those away from the center, according to a Gaussian function.
- *Gaussian Noise: Adds random values to the pixels with a Gaussian probability density function. It does not have a kernel, but it models various noises that can be added to the image.
- Average Filter: Has a kernel with the constant value of $1/(k\times k)$, where k is the kernel size.
- Other transformations that change most pixels in an image using filter-like functions.

Color Changes. This group models changes in the color space of images, which affects all pixels in an image and has differentiable functions. Examples that can be handled by this category include the following:

- Brightness: Adjusts the overall lightness in an image.
- Contrast: Changes the difference between the darkest and lightest colors in an image.
- Color Intensity: Modifies the color balance of an image.
- Sharpness: Changes the sharpness/blurriness of an image.
- Other transformations that manipulate various aspects of the color channels.

Local Pixel Changes. Unlike previous transformations, this group models local changes to a subset of the pixels in an image. However, similar to the Color and Image Filtering categories, transformations in this category also have differentiable functions. Examples that can be handled by this category include the following:

- *Cropout: Removes a square of random size from an image.
- *Dropout: Drops a fraction of bits from an encoded image.
- Pixel Elimination: Removes some rows or columns from an image.
- Other transformations that affect a subset of pixels.

Summary. The above transformations have been grouped based on their effects on images. Other and/or future transformations can easily be modeled by one of these categories or a combination of them. Thus, by training FlexMark on representative transformation samples, along with randomization in their parameters, it can generalize to a wide variety of transformations, including ones that were never seen during training.

3.5 Loss Functions and Model Training

Loss Functions. FlexMark includes three loss functions. The first one is the Mean Absolute Error (MAE), which minimizes the differences between the cover and encoded images and is given by:

$$L_{img} = |(I_{en_j} - I_{co_j})|. \tag{2}$$

To further improve the visual quality, we use ideas from generative adversarial networks (GANs). Specifically, we design a Discriminator to differentiate between images with and without embedded messages, similar in nature to prior works that utilize discriminators to differentiate between real and fake images. Using the outputs from the Discriminator and Encoder, we compute an Adversarial Minimax loss as:

$$L_{adv} = \frac{1}{2}\log Dis(I_{co}) - \frac{1}{2}\log Dis(Enc(I_{co}, I_M)). \tag{3}$$

Finally, to ensure that the retrieved message M' is as close as possible to the original message M, we include the Binary Cross Entropy between M and M' as the third loss function, which is given by:

$$L_M = (y_i \log(\hat{y_i}) + (1 - y_i)\log(1 - \hat{y_i})), \tag{4}$$

where y_i is the i^{th} bit in M, and $\hat{y_i}$ is the i^{th} bit in M'.

The **total loss function** is given by:

$$L_{total} = \mathbb{E}_{I_{co},M}[(1 - \lambda_1 - \lambda_2)L_{adv} + \lambda_1 L_{img} + \lambda_2 L_M], \tag{5}$$

where λ_1, λ_2 are hyper-parameters tuned experimentally. λ_1 is set to 0.7 and λ_2 is set to 0.29 in our experiments.

End-to-end Training of the Model. We train FlexMark end to end. As shown in Figure 3, FlexMark consists of multiple trainable and non-trainable modules. The input image and message first go through the Encoder. The output, the encoded image, then goes through both the Discriminator and one of the transformations, randomly chosen. A differentiable transformation is implemented using multiple layers, whereas a non-differentiable one is implemented using our pre-trained neural network model approximating the non-differentiable transformations, which will pass the gradient through, but it is not trainable by itself.

Finally, the Decoder will extract the retrieved message from the transformed image. In each iteration, the encoded image will go through the Discriminator, which tries to distinguish whether the image is encoded or not. Over time, as the quality of the encoded image increases, it becomes more difficult for the Discriminator to recognize the encoded image. The objective is to make sure the encoded image is similar to the original cover image. Also, the retrieved message should be close to the embedded message.

After the training process is complete, the Encoder model is used to embed messages in images, and the Decoder model is used to extract the embedded messages.

4 EVALUATION

We analyze the performance of FlexMark and compare it against the closest systems in the literature [1, 7] using their publicly available open-source codes. We also conduct an ablation study to analyze the impact of different components of FlexMark.

Transformation	Parameters
JPEG Compression	Quality Factor = [0–100]; **80**
Gaussian Blur	σ=[1,**2**,3]
CropOut	Ratio=[0–1][**(0.2,0.4)**]
DropOut	Probability=[0.1,0.2,**0.3**]
Gaussian Noise	σ=[0.01,0.02,**0.03**]
Pixel Elimination	Ratio=[0.05,**0.1**,0.15]
Saturation	Factor = [0.5,1,**1.5**]
Brightness	Factor = [0.5,1,**1.5**]
Contrast	Factor = [0.55,**0.66**,0.77]
Sharpness	Factor = [0.5,1,**1.5**]
Color Intensity	Factor = [0.5,1,**1.5**]
Average Filter	Kernel = [**3**,5,7]
Median Filter	Kernel = [3,**5**,7]

Table 2: The range of parameter values for each transformation. Bold values are used in testing, while other values are used in training the considered data-hiding methods.

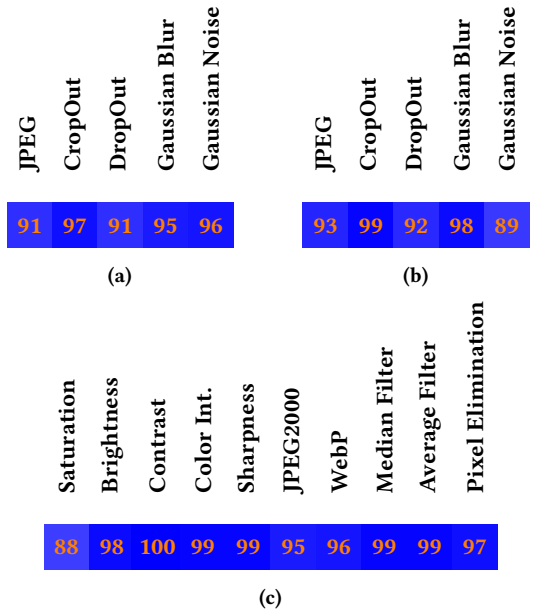


Figure 7: Robustness of FlexMark against: (a) diverse transformations, (b) different parameter values of the transformations, and (c) transformations never seen during training.

4.1 Setup and Dataset

Dataset and Model Training. Similar to prior works [1, 7, 39], we use the COCO dataset [15], which has a large and diverse set of images. We randomly select 15,500 images: 15,000 for training and 500 for testing. We resize and crop the images to 128×128. We train the model for 50 epochs. The batch size for training is 16. We use the Adam optimizer with $\beta_1 = 0.5, \beta_2 = 0.999$, and a learning rate of 0.0002.

We note that FlexMark is trained *only once* on different message sizes and a few representative transformations. During inference, we control the trade-off between capacity and robustness using α. This is to demonstrate the flexibility and applicability of FlexMark to different multimedia applications.

Transformation Parameters. The parameter values of each transformation used in our evaluation are listed in Table 2, which helps in reproducing our results. During training, we randomly select values in the listed ranges for each iteration. We highlight the parameter values used in the test dataset in **bold** font.

Performance Metrics. We consider two main metrics, which were also used in prior works [1, 7]: Capacity and Robustness. Capacity is the number of bits that can be hidden inside a pixel (measured in bits); it is given by $C = |M|$. Robustness is the bit retrieval accuracy, which is the number of matching bits between M' and M divided by $|M|$. As mentioned before, we set $\beta = 0.01$ in all experiments. This was done to ensure the visual quality (in terms of PSNR) of the encoded images is at least 30 dB.

4.2 Performance of FlexMark

Accuracy and Robustness. We analyze the robustness of FlexMark against most practical transformations that we are aware of to demonstrate its generality. Specifically, we train FlexMark on only *five* transformations chosen from different categories, which are: Gaussian Noise, Cropout, Gaussian Blur, JPEG Compression, and Dropout. During training, each distortion is assigned a standardized parameter, e.g., a quality factor of 80 for JPEG Compression. The system performance is initially tested on these five transformations using the same parameters employed during training, as depicted in Figure 7a.

Subsequently, we introduce variability by randomly selecting alternative parameters for the aforementioned distortions. The outcomes of these randomized scenarios are illustrated in Figure 7b. We then extend our evaluation to assess the robustness of FlexMark to previously unseen transformations. In this phase, the parameters are chosen randomly. Figure 7c demonstrates that FlexMark exhibits robustness even against distortions that it was not exposed to during training and is able to retrieve embedded watermarks with high accuracy.

Adaptability. We analyze the adaptability of FlexMark and the impact of different α values on the performance. α controls the trade-off between capacity and robustness. It ranges from 0 to $\sqrt{H \times W}$. Smaller α values correspond to smaller blocks and higher capacity. Smaller blocks need to be up-sampled to the original image size and go through the decoder for message extraction.

In this experiment, we vary the capacity from 4 to 1024 bits. We compute the robustness as the average accuracy across all distortions. We plot the average robustness in Figure 8. The figure also shows the error bars, which are computed as the mean ± the standard deviation. The results indicate that FlexMark is highly accurate in lower bit capacities. This makes FlexMark suitable for applications that need high robustness such as copyright protection and media authentication. As the capacity increases, the robustness decreases, which makes FlexMark suitable for capacity-demanding applications such as covert communication and tele-medicine.

Complexity Analysis and and System Requirements. The Encoder of FlexMark has about 78 million trainable parameters, whereas its Decoder has about 39 million. Recall that FlexMark is

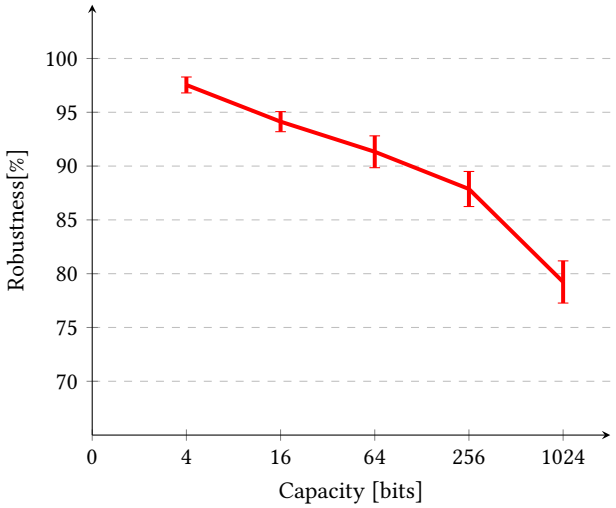


Figure 8: Adaptability of FlexMark, which allows it to control the trade-off between the robustness and capacity to meet the requirements of different applications.

	Message Length (bits)	Time (sec)
Message Embedding	16	0.08
	64	0.08
	256	0.10
Message Extraction	16	0.37
	64	1.43
	256	5.7

Table 3: Execution times for each phase in FlexMark.

trained end to end, meaning that both the Encoder and Decoder are trained together. Training FlexMark requires a single GPU with at least 10 GB of memory. In all of our experiments, we used an NVIDIA Geforce RTX 2080 Ti GPU with 12 GB of memory. On this GPU, training of FlexMark took about 30 hours.

After training, the whole model requires about 300 MB of disk space. The trained Encoder model is used to embed messages in images, and the trained Decoder model is used to extract them.

We measured the average time to embed and extract messages of different sizes, and the results are reported in Table 3. These average times were computed across the 500 images in the testing dataset. As shown in the table, FlexMark takes less than 0.1 sec to embed a message, and it takes up to a few seconds to extract a message. The table also shows that longer messages need more time for extraction. This is because of two reasons. First, the decoder needs to extract messages block by block to extract bits one by one. Thus, if we have a 256-bit message, the decoder is called 256 times. Second, before decoding, the blocks need to be upsampled to the original image size. This upsampling again needs to be done 256 times for a message of size 256. These two reasons make message extraction more time-consuming than message embedding.

4.3 Comparison against State-of-the-Art

We compare FlexMark against the state-of-the-art methods for robust data hiding, which are ReDMark [1] and FIN [7]. To ensure fair comparisons, we tune these methods on our dataset. Additionally, since the capacity in FIN[7] is not adaptable, we train their network for each capacity separately.

Robustness. For fair comparisons, we train all methods on the same transformations: Gaussian Noise, Cropout, Gaussian Blur, JPEG compression, and Dropout. We also configure all methods to produce the same visual quality with PSNR of at least 30 dB.

We assess the robustness of the three data hiding methods against transformations that they have been trained on, as well as on transformations that were not included in the training. Robustness against unseen transformations is crucial, and it shows the practicality of the method. Specifically, we evaluate the robustness against 15 different transformations: 5 were seen during training and 10 were not. The results are summarized in Figure 9, which are color-coded to facilitate comparisons: darker cells mean higher accuracy. The number in each cell of the table represents the average accuracy for the considered watermarking method and message size (in the corresponding row) and applied transformation (in the corresponding column). Notice that each cell in the table represents a whole separate experiment, thus, this color-coded table presents the results of numerous experiments in a compact form.

Figure 9 shows that, unlike the state-of-the-art data hiding methods, FlexMark consistently results in high accuracy across all transformations, including the ones that were not included in the training dataset. This is because FlexMark is designed to consider the fundamental aspects of all different transformations, as described in §3.4, and it does not need to be fine-tuned on specific transformations. This is unlike FIN and ReDMark which were tuned on a specific set of transformations, which made them fail for some transformations such as Gaussian Blur and Median Filter.

We note that some transformations only affect scattered pixels, like DropOut, while others, such as Gaussian blur, affect the entire image. Various components of FlexMark, e.g., message expansion, double embedding, and self-attention, enable it to extract messages even in the presence of distortions. This makes FlexMark more robust than other methods, which fail in similar situations.

Variable Capacity. We compare the capacity of the considered data-hiding methods. Both FIN and ReDMark can support messages of various sizes. For images of size 128×128 pixels, the maximum message size for both of them is 256 bits. The results in Figure 9 show that FlexMark substantially outperforms ReDMark and FIN, by offering much higher bit accuracy in each capacity level and consistently providing higher robustness against all transformations, especially for large message sizes and unseen transformations.

4.4 Ablation Study

We assess the performance contribution of various components of FlexMark, including the transformation module, double embedding of the input message, and using a discriminator in the loss function. We start with a base model without any of the aforementioned components. Then, we add components incrementally and measure their impact. The results are summarized in Figure 10,

Method	Message Length	JPEG	CropOut	DropOut	Gaussian Blur	Gaussian Noise	Saturation	Brightness	Contrast	Color Int.	Sharpness	JPEG2000	WebP	Median Filter	Average Filter	Pixel Elimination
ReDMark [1]	16	85	97	70	50	100	83	96	95	95	100	100	100	50	50	92
FIN [7]		74	99	97	55	100	85	79	85	84	90	69	69	55	70	100
FlexMark (ours)		91	97	91	95	96	86	93	99	97	97	88	94	96	98	94
ReDMark [1]	64	81	97	68	50	100	81	94	94	93	100	98	100	50	50	90
FIN [7]		70	99	96	55	100	83	77	85	82	87	68	69	55	59	100
FlexMark (ours)		87	96	91	83	94	82	94	97	97	98	81	87	91	97	95
ReDMark [1]	256	62	83	63	52	97	76	91	90	91	91	96	80	50	50	87
FIN [7]		69	99	94	50	100	80	77	84	78	84	68	69	54	50	100
FlexMark (ours)		85	95	88	87	81	80	77	91	97	96	81	80	83	93	95

Figure 9: Comparing FlexMark against state-of-the-art, ReDMark [1] and FIN [7], on 15 transformations: the left 5 transformations were seen during training and the other 10 were not. Darker colors mean higher accuracy. White color represents no accuracy, i.e., totally random results. Three message lengths (16, 64, and 256 bits) are considered.

for three sample transformations: Gaussian Noise, JPEG Compression, and Gaussian Blur. The results indicate that our base model already offers some resilience to different transformations, partly attributed to our message expansion method. The transformation module notably has the greatest impact on improving the bit accuracy because it helps the model become familiar with distortions during training. The discriminator loss comes next, making sure that the protected images stay similar to the original ones. Finally, the double embedding of the input message further improves the bit accuracy, because it allows the neural network model to consider the characteristics of the message and the cover image at two different locations: in the early layers where low-level features are considered, and in later layers where conceptual and high-level features are emphasized.

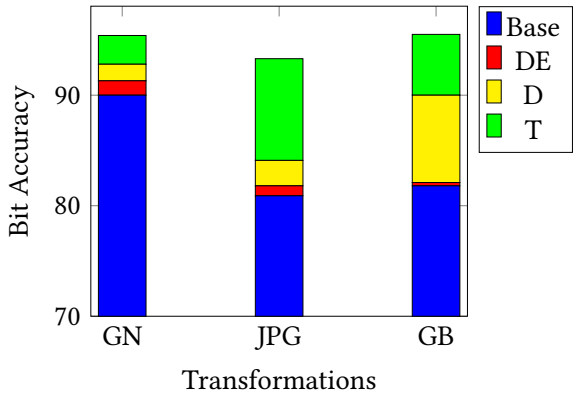


Figure 10: Ablation study. Performance impact of various components: Double Embedding (DE), using Discriminator in the loss function (D), and transformation module (T). Results are shown for three sample transformations: Gaussian Noise (GN), JPEG Compression (JPG), and Gaussian Blur (GB).

5 CONCLUSIONS

We presented FlexMark, an adaptable, high-capacity, and robust watermarking method for images. FlexMark can be customized for different applications by controlling its parameter, α. The robustness of FlexMark is achieved by modeling the essential aspects of image transformations and training the model on them, whereas the high capacity is enabled by its Encoder design that utilizes multiple ideas such as double embedding of the input message. Through extensive experiments, we showed that FlexMark is adaptable and can achieve high accuracy and robustness against a large variety of transformations such as lossy compression, image filtering, pixel eliminations, and color changes. In addition, we demonstrated that FlexMark offers robustness against many transformations not seen during training, which makes it more practical than current methods in the literature. We conducted an ablation study to analyze the performance contributions of various components of FlexMark. Finally, we showed that FlexMark outperforms the closest methods in the literature in terms of robustness and capacity, while offering adaptability to different applications.

REFERENCES

[1] Mahdi Ahmadi, Alireza Norouzi, Nader Karimi, Shadrokh Samavi, and Ali Emami. 2020. ReDMark: Framework for residual diffusion watermarking based on deep networks. *Expert Systems with Applications* 146 (2020), 113157.

[2] Hazem Munawer Al-Otum and Mouaz Ibrahim. 2021. Color image watermarking for content authentication and self-restoration applications based on a dual-domain approach. *Multimedia Tools and Applications* 80, 8 (2021), 11739–11764.

[3] Lori Cameron. 2017. Hackers' latest weapon: Steganography: IEEE Computer Society. https://www.computer.org/publications/tech-news/research/how-steganography-works

[4] Deepshikha Chopra, Preeti Gupta, Gaur Sanjay, and Anil Gupta. 2012. LSB based digital image watermarking for gray scale image. *IOSR journal of Computer Engineering* 6, 1 (2012), 36–41.

[5] Xintao Duan, Kai Jia, Baoxia Li, Daidou Guo, En Zhang, and Chuan Qin. 2019. Reversible image steganography scheme based on a U-Net structure. *IEEE Access* 7 (2019), 9314–9323.

[6] Ersin Elbasi and Volkan Kaya. 2018. Robust medical image watermarking using frequency domain and least significant bits algorithms. In *2018 International Conference on Computing Sciences and Engineering (ICCSE)*. IEEE, 1–5.

[7] Han Fang, Yupeng Qiu, Kejiang Chen, Jiyi Zhang, Weiming Zhang, and Ee-Chien Chang. 2023. Flow-Based Robust Watermarking with Invertible Noise Layer for Black-Box Distortions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 5054–5061.

[8] Zhenyu Guan, Junpeng Jing, Xin Deng, Mai Xu, Lai Jiang, Zhou Zhang, and Yipeng Li. 2022. DeepMIH: Deep Invertible Network for Multiple Image Hiding. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[9] Daniel Jiwoong Im, Chris Dongjoo Kim, Hui Jiang, and Roland Memisevic. 2016. Generating images with recurrent adversarial networks. *arXiv preprint arXiv:1602.05110* (2016).

[10] Junpeng Jing, Xin Deng, Mai Xu, Jianyi Wang, and Zhenyu Guan. 2021. HiNet: deep image hiding by invertible network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4733–4742.

[11] Inas Jawad Kadhim, Prashan Premaratne, Peter James Vial, and Brendan Halloran. 2019. Comprehensive survey of image steganography: Techniques, Evaluations, and trends in future research. *Neurocomputing* 335 (2019), 299–326.

[12] Ton Kalker, Geert Depovere, Jaap Haitsma, and Maurice JJJB Maes. 1999. Video watermarking system for broadcast monitoring. In *Security and Watermarking of Multimedia contents*, Vol. 3657. SPIE, 103–112.

[13] Songul Karakus and Engin Avci. 2020. A new image steganography method with optimum pixel similarity for data hiding in medical images. *Medical Hypotheses* 139 (2020), 109691.

[14] N Senthil Kumaran and S Abinaya. 2016. Comparison analysis of digital image watermarking using DWT and LSB technique. In *2016 International Conference on Communication and Signal Processing (ICCSP)*. IEEE, 0448–0451.

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[16] Lianshan Liu, Li Tang, and Weimin Zheng. 2022. Lossless Image Steganography Based on Invertible Neural Networks. *Entropy* 24, 12 (2022), 1762.

[17] Yang Liu, Mengxi Guo, Jian Zhang, Yuesheng Zhu, and Xiaodong Xie. 2019. A novel two-stage separable deep learning framework for practical blind watermarking. In *Proceedings of the 27th ACM International conference on multimedia*. 1509–1517.

[18] Shao-Ping Lu, Rong Wang, Tao Zhong, and Paul L Rosin. 2021. Large-capacity image steganography based on invertible neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10816–10825.

[19] Xiyang Luo, Ruohan Zhan, Huiwen Chang, Feng Yang, and Peyman Milanfar. 2020. Distortion agnostic deep watermarking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13548–13557.

[20] Pratap Chandra Mandal, Imon Mukherjee, Goutam Paul, and BN Chatterji. 2022. Digital image steganography: A literature survey. *Information Sciences* (2022).

[21] Nour Mohamed, Mohammed Baziyad, Tamer Rabie, and Ibrahim Kamel. 2020. L* a* b* color space high capacity steganography utilizing quad-trees. *Multimedia Tools and Applications* 79, 33 (2020), 25089–25113.

[22] Shabir A Parah, Javaid A Sheikh, Farhana Ahad, Nazir A Loan, and Ghulam Mohiuddin Bhat. 2017. Information hiding in medical images: a robust medical image watermarking system for E-healthcare. *Multimedia Tools and Applications* 76, 8 (2017), 10599–10633.

[23] Goutam Paul, Sanjoy Kumar Saha, Debanjan Burman, et al. 2020. A PVD based high capacity steganography algorithm with embedding in non-sequential position. *Multimedia Tools and Applications* 79, 19 (2020), 13449–13479.

[24] Zhenxing Qian, Hang Zhou, Weiming Zhang, and Xinpeng Zhang. 2017. Robust steganography using texture synthesis. In *Advances in Intelligent Information Hiding and Multimedia Signal Processing*. Springer, 25–33.

[25] Arkadip Ray and Somaditya Roy. 2020. Recent trends in image watermarking techniques for copyright protection: a survey. *International Journal of Multimedia Information Retrieval* 9, 4 (2020), 249–270.

[26] Maria L Ricci. 2010. United States Department of Justice. https://www.justice.gov/sites/default/files/opa/legacy/2010/06/28/062810complaint2.pdf Accessed : 1-1-2023.

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

[28] Haichao Shi, Jing Dong, Wei Wang, Yinlong Qian, and Xiaoyu Zhang. 2017. SSGAN: secure steganography based on generative adversarial networks. In *Pacific Rim Conference on Multimedia*. Springer, 534–544.

[29] Ajib Susanto, Eko Hari Rachmawanto, Christy Atika Sari, et al. 2018. A robust non-blind image watermarking method using 2-level HWT-DCT. In *2018 International Seminar on Application for Technology of Information and Communication*. IEEE, 304–308.

[30] Gandharba Swain. 2019. Very high capacity image steganography technique using quotient value differencing and LSB substitution. *Arabian Journal for Science and Engineering* 44, 4 (2019), 2995–3004.

[31] Jinyuan Tao, Sheng Li, Xinpeng Zhang, and Zichi Wang. 2018. Towards robust image steganography. *IEEE Transactions on Circuits and Systems for Video Technology* 29, 2 (2018), 594–600.

[32] Denis Volkhonskiy, Boris Borisenko, and Evgeny Burnaev. 2016. Generative adversarial networks for image steganography. (2016).

[33] Denis Volkhonskiy, Ivan Nazarov, and Evgeny Burnaev. 2020. Steganographic generative adversarial networks. In *Twelfth International Conference on Machine Vision (ICMV 2019)*, Vol. 11433. International Society for Optics and Photonics, 114333M.

[34] Pin Wu, Yang Yang, and Xiaoqiang Li. 2018. Image-into-image steganography using deep convolutional network. In *Pacific Rim Conference on Multimedia*. Springer, 792–802.

[35] Pin Wu, Yang Yang, and Xiaoqiang Li. 2018. Stegnet: Mega image steganography capacity with deep convolutional network. *Future Internet* 10, 6 (2018), 54.

[36] Youmin Xu, Chong Mou, Yujie Hu, Jingfen Xie, and Jian Zhang. 2022. Robust Invertible Image Steganography. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7875–7884.

[37] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. 2019. Self-attention generative adversarial networks. In *International conference on machine learning*. PMLR, 7354–7363.

[38] Yi Zhang, Chuan Qin, Weiming Zhang, Fenlin Liu, and Xiangyang Luo. 2018. On the fault-tolerant performance for a class of robust image steganography. *Signal Processing* 146 (2018), 99–111.

[39] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. 2018. Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*. 657–672.