Expert Systems With Applications 222 (2023) 119841

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Evolving Generative Adversarial Networks to improve image steganography

Alejandro Martín [a,*], Alfonso Hernández [b], Moutaz Alazab [c], Jason Jung [d], David Camacho [a]

[a] Departamento de Sistemas Informáticos, Universidad Politécnica de Madrid, Spain
[b] Department of Computer Science, Universidad Rey Juan Carlos, Spain
[c] Department of Intelligence Systems Department, AlBalqa Applied University, Amman, Jordan
[d] Department of Computer Engineering, Chung-Ang University, Korea

ARTICLE INFO

Keywords:
Steganography
Generative Adversarial Networks
GAN
Genetic algorithm

ABSTRACT

Images have been repeatedly used as the perfect environment to hide information through the use of steganography techniques. Whether messages, documents or even other images, the bitmap of an digital picture provides a place where hidden data can be embedded without human notice. So far, a plethora of steganography methods can be found in the state-of-the-art literature, together with steganalysis techniques, devoted to detect the presence of hidden information in files. Recent steganography techniques rely on Convolutional Neural Networks, trying to embed as information as possible while minimising visual changes in the image. Following this trend, this article tries to demonstrate that a Generative Adversarial Network (GAN) can be used to improve the ability of a spatial domain steganalysis method and to insert secret information with minimal image alteration. Through a training process, the GAN learns how to adapt an image to later introduce a message using the Least Significant Bit steganography algorithm. The results evidence that the approach is successful at avoiding detection by a state-of-the-art Deep Learning steganalysis architecture.

1. Introduction

Over the years, numerous methods have been proposed to hide information and maintain secure communication channels (Amirtharajan & Rayappan, 2013). With the invention of computers and digital resources, a new scenario started where almost any file can be used to hide information visible to human eyes. Images, audio files, a PDF or even the control-flow diagram of a programme after editing its binary code can be used to embed messages or files through steganography techniques (El-Khalil & Keromytis, 2004). In contrast to cryptography, which refers to techniques and algorithms where the message is changed in order to make it unintelligible for unwanted readers, steganography is the art of making information invisible to avoid being detected by a third party (Kadhim, Premaratne, Vial, & Halloran, 2019). To achieve maximum protection of a given information piece, both techniques can be used in conjunction, embedding an encrypted messaged into an image through a steganographic algorithm.

In parallel, researchers and companies have also developed steganalysis techniques (Karampidis, Kavallieratou, & Papadourakis, 2018), aiming to detect the presence of hidden messages within another document. Over the years, a large number of steganalysis techniques has been proposed, analysing different elements of the cover medium where the hidden message is embedded. However, the adoption of Deep Learning architectures by the majority of recent approaches for steganography has led researchers to also leverage these architectures to build steganalysis tools and to reveal signs of manipulation. These architectures are trained with the goal of minimising the changes required in the cover medium to include new hidden information while minimising the visual change (Kodovsky, Fridrich, & Holub, 2011; Xu, Wu, & Shi, 2016).

One of the most preferred cover mediums for embedding hidden messages are images. A high-resolution picture provides a perfect place to introduce new information, altering the image without human notice.

In contrast to other types of files, it is easy to apply modifications (changing the value of a pixel of one specific channel), which hampers the application of manipulation detection methods. Thus, images are widely used in steganography tools. In this scenario, Convolutional Neural Networks (CNNs) have become the leading approach given their possibilities to deal with images. However, these models are vulnerable to *adversarial attacks*, which weaken their robustness (Finlayson et al., 2019). In image classification models, this can be done by adding

* Correspondence to: Departamento de Sistemas Informáticos, Universidad Politécnica de Madrid, Campus Sur de la UPM, C. de Alan Turing, s/n, 28031 Madrid, Spain.
E-mail addresses: alejandro.martin@upm.es (A. Martín), alfonso.h.silva@gmail.com (A. Hernández), m.alazab@bau.edu.jo (M. Alazab), j2jung@gmail.com (J. Jung), david.camacho@upm.es (D. Camacho).

https://doi.org/10.1016/j.eswa.2023.119841
Received 23 September 2022; Received in revised form 22 February 2023; Accepted 7 March 2023
Available online 10 March 2023

small changes to the input image in order to provoke an error in the classification model.

This work proposes a new *Generative Adversarial Network* (GAN) model able to perform adversarial attacks against a steganalysis model of the state of the art. This GAN model involves a discriminator, a steganalysis model based on a CNN architecture from the state of the art literature, and a generator, which is also composed of convolutional layers and has been designed and presented in this work with the goal of preparing an image (i.e. it applies a series of small changes into the input image) for robust steganography and hampering the use of steganalysis models.

In order to address the definition of a generator architecture capable of performing such adaptation of the image before the steganographic message is embedded, this work explores different techniques based on evolutionary computation to find the most adequate topology. The final implementation involves a genetic algorithm which evolves a population of generators following a multi-objective scheme: one objective involves finding a generator able to maximise the error rate in the discriminator while a second objective involves minimising the changes introduced in the original image.

This work is structured as follows: Section 2 reviews the existing work; Section 3 describes the designed GAN architecture; Section 4 describes the genetic algorithm implemented for evolving the GAN architecture; Section 5 describes the experiments performed and the results; Section 6 provides a discussion of the results and, finally, Section 7 shows the conclusions extracted and future work.

2. Background and related work

In this section, a review of existing work on image steganography, steganalysis, generative adversarial networks and deep learning based approaches is provided.

2.1. Image steganography techniques

Image steganography refers to techniques able of introducing secret information (the steganographic message) within an image in a way that it cannot be discovered by a third party. In contrast to cryptography, which performs operations on the information to make it unreadable, steganography tries to make it invisible. Steganography methods can be classified into 2 different embedding domains: spatial and transformation (Cheddad, Condell, Curran, & Mc Kevitt, 2010).

2.1.1. Spatial domain

These algorithms manipulate the image pixels' bits to embed the steganographic message. The simplest method is the *Least Significant Bit* (LSB) algorithm (Neeta, Snehal, & Jacobs, 2006), which modifies the less significant bit of a certain number of pixels of the image in order to embed the message. The first LSB based methods were called LSB replacement (LSB-r) and operated by replacing the less significant bit of subset of the image pixels with the steganographic message. Other methods are LSB matching (LSB-m), which modify certain pixels by increasing or decreasing the value of the LSB in order to make the bits in the image and the steganographic message match.

Methods based on the *Pixel Value Difference* (PVD) have also strong relevance. These methods were proposed for the first time by Wu and Tsai (2003) and use an analysis of the difference between adjacent pixels to decide how much information can be embedded in each pixel. PVD based methods allows higher payload capacities but are easier to detect with histogram analysis approaches. Variations of these methods try to minimise this issue, combining LSB and PVD (Wu, Wu, Tsai, & Hwang, 2005), or applying PVD to different image blocks (Yang, Weng, Tso, & Wang, 2011), among others. In addition to these methods, another important approach is *Exploit Modification Direction* (EMD), presented for the first time by Zhang and Wang (2006). This approach sacrifices payload capacity to obtain better image fidelity. This work focuses on spatial domain methods due to its strong presence in state-of-the-art literature related to steganography techniques.

2.1.2. Frequency or transformation domain

Frequency domain methods are based on the idea of transforming the image to the frequency domain, to embed the secret message and to transform the image again to the spatial domain. These methods are robust and hard to be detected. In the literature, it is possible to find methods based on different transformations, such as the *Discrete Cosine Transformation* (DCT) (Watson et al., 1994), *Discrete Wavelet Transformation* (DWT) (Stanković & Falkowski, 2003), and those based on the *Discrete Fourier Transformation* (DFT) (Paulson, 2006).

2.2. Steganography evaluation methods

The previous section summarises the two main categories of steganography methods. Within these two main categories, there are varied algorithms with different pros and cons. In order to evaluate the quality of an steganographic method, different methods have been proposed in the literature. According to the classification raised by Hussain, Wahab, Idris, Ho, and Jung (2018), the following categories can be defined:

- Visual methods: There are different metrics to asses if the image quality is not reduced once the steganographic message is introduced. The simplest one involves computing the Euclidean Distance between every pixel of both images. Other metrics explored in the state-of-the-art literature are Mean Quadratic Error, the Image Quality Index, Structural Similarity Image Quality Assessment, Image Fidelity or the Mean Difference (see Table 1).
- Payload capacity: measures the size of the message that can be embedded. It is computed as shown in Eq. (1).

$$PayloadCapacity = \frac{No.of\,Embedded\,Bits}{Heigh \times Width} \tag{1}$$

- Security: measures the ability to avoid a detection attack, consisting on extracting information from the image that helps to infer if it contains hidden information. These include statistical features based attacks (Dumitrescu, Wu, & Wang, 2002; Lee, Westfeld, & Lee, 2007), and non-structural based attacks (Pevny, Bas, & Fridrich, 2010).
- Robustness: this is an evaluation method specially important in transformation-based domain methods. It measures the method's capacity to keep the message unaffected after some processing techniques are applied to the image, such as noise addition, rotation, etc.
- Computational complexity: measures the execution time and the number of operations to embed and extract the steganographic message. It is preferable to choose methods requiring low execution time.

2.3. Image steganalysis

Steganalysis refers to those methods dedicated to detect the presence of steganographic messages inside an image. There are many different steganalysis methods, and many different taxonomies. One of the most extended taxonomies was presented by Karampidis et al. (2018), which distinguishes between *visual*, *statistical*, *non structural*, *spread spectrum*, *transformation domain methods* and *blind* or *universal* methods. The following describes each of these categories:

2.3.1. Visual steganalysis

This is the simplest approach, and consists on detecting the steganographic message relying only on the human eye. Any steganographic method whose embedded messages are detected using this method can considered useless. As explained in previous section, to measure the visual quality of *stego* images there exist different metrics available (Table 1).

Table 1
Metrics for the visual analysis of the images. In the table, H is the image height, W is the image width, X and Y are the pixel values, X′ and Y′ are the pixels mean and N is the number of measures.

Metric	Equation
Mean quadratic error	$MSE(X,Y) = \frac{1}{H\times W}\sum_{i=1}^{H\times W}(X_i - Y_i)^2$
Image quality Index (Index Q) (Wang & Bovik, 2002)	$Q = \frac{4\times(\theta_{XY})\times X''\ \times\ Y''}{((\theta_X)^2+(\theta_Y)^2)[(X'')^2+(Y'')^2]}$ $X'' = \frac{1}{N}\sum_{J=1}^{N} X_J$ $Y'' = \frac{1}{N}\sum_{J=1}^{N} Y_J$ $(\theta_X)^2 = \frac{1}{N-1}\sum_{J=1}^{N}(X_J - X'')^2$ $(\theta_Y)^2 = \frac{1}{N-1}\sum_{J=1}^{N}(Y_J - Y'')^2$ $\theta_{XY} = \frac{1}{N-1}\sum_{J=1}^{N}(X_J - X'')(Y_J - Y'')$
Structural similarity Image quality Assessment (Wang, Bovik, Sheikh, & Simoncelli, 2004)	$SSIM(X,Y) = (\frac{(2\times X'\ \times\ Y'+K_1)(2\times M_{XY}+K_2)}{(M_X^2+M_Y^2+K_2)\times((X')^2+(Y')^2+K_1}$) X′ e Y′ are the pixels mean, M_X and M_Y are the variance arrays; M_{XY} are the co-variance arrays and K_1 y K_2 are constants
Image fidelity	$FI = 1 - (\frac{\sum_{i=1}^{H\times W}(X_i\times Y_i)^2}{\sum_{i=1}^{H\times W} X_i^2}$)
Mean difference (Kumar & Rattan, 2012)	$DM(X,Y) = \frac{1}{H\times W}\sum_{i=1}^{H\times W}(X_i - Y_i)$

2.3.2. Statistical steganalysis

This category refers to a wide number of different methods, where the main idea is to inspect some statistical features extracted from the suspicious image. These methods are mainly based on histogram analysis (Ker, 2005), which seeks for the difference in the histogram between cover and stego images; other approaches are based on chi-square attacks (Westfeld & Pfitzmann, 1999), initially developed to detect LSB steganography.

2.3.3. Non structural steganalysis

These methods extract features from the image and use them to feed a machine learning classifier, i.e. a support vector machine (SVM). The most representative model from this family is the Pixel Adjacency Matrix (SPAM) (Zhang, Ping, Xu, & Wang, 2014) and the Spatial Rich Model (SRM) (Fridrich & Kodovsky, 2012).

2.3.4. Spread spectrum and transformation domains

These two categories seek to find messages embedded applying methods which modify the image spreading it as a signal to introduce small variations. Methods propose using this type of method allow to retrieve hidden data using a key (Marvel, Boncelet, & Retter, 1999).

2.3.5. Universal methods

These algorithms are designed to find stenographic messages embedded using an unknown steganographic method, making them especially interesting. These methods are usually *non structural*, and involve machine learning techniques to classify between cover and stego images. Some works employ other techniques such as dimensional reduction (Lu, Liu, & Luo, 2014) or coefficients correlation (Zong, Liu, & Luo, 2012). Nowadays, the most employed methods in (universal) steganalysis are based on deep learning, specially those based on convolutional neural networks (CNN).

2.4. Convolutional neural networks

Convolutional Neural Networks (CNN) are a specific type of neural network based on convolutional operations. They were first presented in 1990 by LeCun et al. (1990). Two decades later, on 2012, Krizhevsky, Sutskever, and Hinton (2012) presented the model *AlexNet*, which outperforms any other model in all categories of the *imagenet* competition. Its success was not only related to its architecture but to 2 factors: the data pre-processing that was performed before the training process to increase the training set size. The second one is the training strategy, which consists on splitting the training process on 2 different flows and executing each one in a different GPU. Furthermore, CNNs can be stacked through an ensemble approach, improving the results of isolated models (Huertas-Tato, Martín, Fierrez, & Camacho, 2022).

CNNs have become the most widely used algorithm for image analysis tasks, such as image classification and image segmentation, among others. Image steganalysis is not an exception and most of the actual image steganalysis methods are based on CNNs. In 2015, Qian, Dong, Wang, and Tan (2015) presented the first model based on Deep Learning, achieving as good results as other state of the art models relying on classical methods, such as *Spatial Rich Model* (SRM) (Fridrich & Kodovsky, 2012). In 2017, Wu, Zhong, and Liu (2019) presented a new type of normalisation layer called *shared normalisation* specifically designed for steganalysis models. In the same year, Ye, Ni, and Yi (2017) presented the *yenet* model, with a new activation function called *Truncated Linear Unit* (TLU). This is the steganalysis model that was chosen for this research due to its good results in several benchmark datasets of the state-of-the-art literature, including *BOSSBase*. Other steganalysis methods based on CNNs are the one presented by Ke, Ming, and Daxing (2018), that employs a multi-column CNN (MCCNN) to allow different input size and resolution, or the one presented by Zhang, Zhu, Liu and Liu (2019), which uses small filters and a special type of *pooling* (He, Zhang, Ren, & Sun, 2015).

2.5. GANs

Generative Adversarial Networks (GANs) were presented by Goodfellow et al. (2020). They are composed of 2 different networks, a *generator* which generates samples from a noise vector and a *discriminator* which is a classification model. This type of model is based on *game theory* where both networks compete with each other. The *generator* tries to fool the *discriminator* while this one try to discern fake samples (generated by the *generator*) from real ones. In 2015, Radford, Metz, and Chintala (2015) presented a model called *Deep Convolutional Generative Adversarial Network* (DCGAN), which replaces dense layers with transpose convolutional layers to make a model capable of creating realistic images.

After that, several variants of GAN models have been proposed. These variants are valid for standard GANS and for convolutional GANs, and among them it is possible to find the *Conditional GAN* (CGAN), presented by Mirza & Osindero (Mirza & Osindero, 2014), which takes as input a batch of noise and a label to generate a sample from the desired class. Other variant was presented by Chen et al. (2016). It is called *InfoGAN* and it employs 2 discriminators to improve CGAN results. Other improvement of CGAN models is the one called *Auxiliary Classifier GAN* (ACGAN), that also gives the probability of the sample of belonging to each class but uses a single discriminator network.

In steganography and steganalysis there are some works that employ GANs. The model SSGAN, presented by Shi, Dong, Wang, Qian, and Zhang (2017), was one of the first models of this kind and it creates *cover* images using the generator and evaluates them using the discriminator. Tang, Tan, Li, and Huang (2017) designed the ASDL-GAN, which apply distortions to cover images to, once the message is embedded, its undetectability is maximised.

In 2018, Yang, Liu, Kang, Wong, and Shi (2018) used a GAN to find the pixels where is more secure to hide the message. Wang, Gao, Wang, Qu, and Li (2018) designed the model SSTeGAN which can generate a stego image using a steganographic message as seed. Zhang, Dong and Liu (2019) designed the ISGAN that can minimise the divergence between cover and stego images, and Zhang, Cuesta-Infante, Xu and Veeramachaneni (2019) presented the StegoGAN, which can embed a message in an image and recover it. Naito and Zhao (2019) propose the use of GANs to maximise the naturalness of the cover medium where the message is embedded.

Volkhonskiy, Nazarov, and Burnaev (2020) focus on deceiving a steganography technique through the modification of the cover image. The authors use the term container for this purpose. Similarly, other researchers have proposed a steganography without embedding method where the message is embedded through a noise vector and a generator (Hu, Wang, Jiang, Zheng, & Li, 2018). AdvGAN is another approach where the GAN is used to learn a steganography scheme involving a steganographic encoder and a steganographic decoder (Li, Fan, & Liu, 2021). Finally, a coverless method which proposes a GAN to encode messages into the cover image was proposed by Qin et al. (2020). In contrast to all these works, our research implements an evolutionary scheme to increase diversity of the population of generators and prepares the cover image to embed the message, minimising the possibility of being detected.

2.5.1. Evolving deep learning models

In the state-of-the-art literature, several authors have proposed various methods to automatically decide the most appropriate topology and hyperparameters of deep learning architectures. One possibility is to use evolutionary computation techniques, such as genetic algorithms. There are several algorithms of this kind that have been used to create and/or optimise different neural networks. The first approach of this kind was presented by Yao (1999). After that, evolutionary computation techniques have been employed to design different aspects of neural networks, such as the number of neurons at each layer (Leung, Lam, Ling, & Tam, 2003) or the number of layers and the number of neurons per layer (Gascón-Moreno, Salcedo-Sanz, Saavedra-Moreno, Carro-Calvo, & Portilla-Figueras, 2013), among others. Evolutionary computation methods have been also used to train the network (Abdalla, 2014).

2.5.2. Evolutionary computation in steganography and steganalysis

In steganography and steganalysis, only a few works use evolutionary computation techniques. Tseng, Chan, Ho, and Chu (2008) proposed a genetic algorithm to find the most adequate pixels to embed the steganography message, and Ghasemi, Shanbehzadeh, and Fassihi (2010) proposed a similar method but dividing the images in pieces of 8×8 pixels. In 2014, Kanan and Nazeri (2014a) proposed a genetic algorithm that looks for the best direction to embed the message and that achieved a high number of optima solutions. There are also methods for the transformation domain, such as the one proposed in 2006 by Fard, Akbarzadeh-T, and Varasteh-A (2006), which embeds the message using the DWT, the one presented in 2017 by Miri and Faez (2017) which modifies the coefficients while minimising visual changes or a method based on integer wavelet transform using genetic algorithm (Sabeti, Sobhani, & Hasheminejad, 2022).

Most of the methods based on evolutionary computation techniques are used as feature extractors previous to a supervised classification model, such as the work of Geetha and Kamaraj (2010) and Ramezani and Ghaemmaghami (2010). Other work, presented by Visavalia and Ganatra (2014), uses a genetic algorithm to create a set of decision rules to classify the images between cover and stego.

3. A GAN architecture for steganography

A Generative Adversarial Network integrates two components called generator and discriminator. The generator is trained with the objective of creating new examples to deceive the *discriminator*. Simultaneously, the discriminator learns how to not being deceived by the generator, that is to say, a classification task to distinguish between real and fake inputs. In this research, a GAN architecture to improve steganography is proposed, where the learning process pursues a set of modifications that once applied to the bitmap lead to a more robust stenographic method. In other words, through a series of specific small modifications in the bitmap, the GAN prepares an input image to later introduce a steganographic message which will deceive the steganalysis model, represented by the discriminator. In addition, the image should be modified as less as possible, being indistinguishable to the naked eye.

In this section it is described the architecture of the generator, the discriminator, the training strategy and the module to embed the steganographic message into the generated images.

3.1. Generator

The generator consists of a convolutional neural network (CNN) which takes a fixed-size image as input and returns an image with the same dimensions. This network receives a *cover* image as input (i.e, an image without a steganographic message). To add some random factor to the image modification, another channel composed of Gaussian noise with mean 0 and variance 1 is included. The CNN input therefore will be composed of 2 channels and will return 1 channel representing an imagen preserving the same dimensions as the input image.

Given that the goal is to return an image as similar as possible to the one used as input, an scheme that enlarges and reduces the size of the input image is used. First the input image dimensions are increased using transpose convolutions. Then, once the image has been enlarged, a mirror architecture is used to reduce it to its original size. This avoids data loss while it requires more computational resources.

As one can see in Fig. 1 the proposed architecture has 5 transpose convolutional layers which increase the input size from 256×256 to 528×528. In addition, the number of filters increase from 2 to 16 in the first layer and to 32 in the third layer. After this, there are six convolutional layers which decrease the image size to 256×256, its original size. *Leaky Relu* is used as activation function in all layers except in the last one, which use *tanh* activation function, as recommended in the literature (Radford et al., 2015). Finally, while layers 6 and 7 have *kernel size* 5, the others have *kernel size* 3.

3.2. Discriminator

For the discriminator, a steganalysis model based on a CNN architecture from the state of the art was selected. The chosen model is *yenet* (Ye et al., 2017). This CNN model will be pre-trained using the same dataset that in all the experiments performed in this work. This will allow to extend this work to other steganalysis models.

3.3. Steganographic module

Once the generator has modified the input image, the steganographic message can be embedded. For that purpose, it is necessary to integrate a new module between the generator and the discriminator running in the same GPU computing environment. The GAN model was implemented using *Tensorflow*, so the module will implement the steganographic algorithm using *Tensors*.

3.4. Training strategy

The GAN architecture together with the training process flow is displayed in Fig. 3. It must be noted that the generator is trained pursuing 2 different objectives. The first one is to keep the modified image as similar to the original image as possible. To that end, *cosine similarity* measure is used as loss value. The *cosine similarity* was chosen since, in contrast to other metrics, its provides a reliable metric based on orientations, and that can be safely applied where magnitudes are not an issue (Xia, Zhang, & Li, 2015). In the scenario proposed in this research, the comparison is performed between images where features (i.e. a pixel) is defined by a bounded range of values. Besides, cosine similarity has already been successfully integrated in GAN architectures due to its flexibility and efficiency (Zhan et al., 2022), but also in other tasks such as contrastive learning (Chen et al., 2022). This function is shown in Eq. (2). The original image is considered as the label while the modified image is used as the predicted class. The loss value consists of

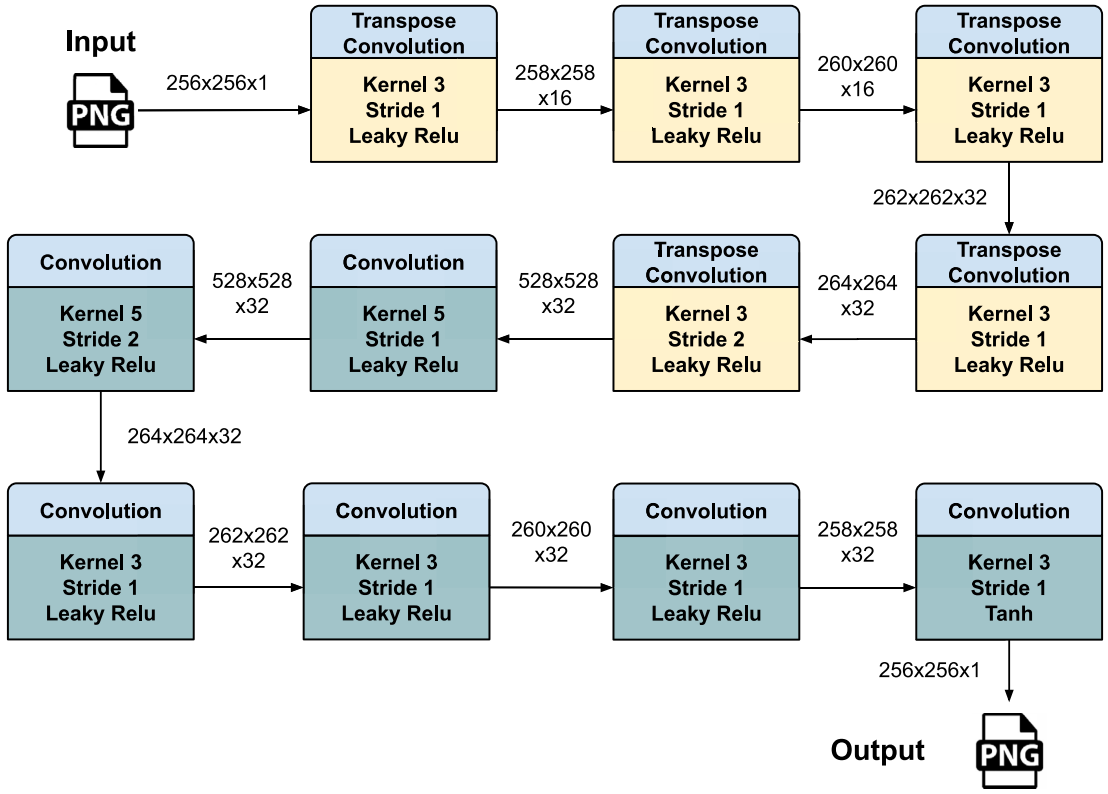

Fig. 1. Diagram showing the layers composing the generator and discriminator modules of the Generative Adversarial Network designed.

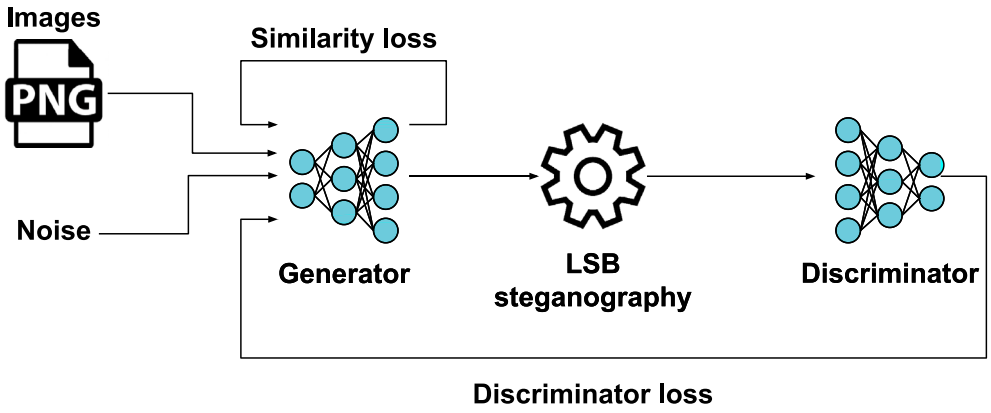

Fig. 2. General overview of the GAN modules and training process.

a matrix with the same dimensions that the output (and input) matrix. This loss propagates through the generator updating the weights.

$$SIM(X,Y) = \frac{\langle X,Y\rangle}{\|X\| * \|Y\|} \tag{2}$$

The second stage consists on training the generator to deceive the discriminator. Starting from the modified image, the first step involves embedding the steganographic message using the module explained above. Our goal is to generate images that, once the steganographic message is embedded, they are classified by the discriminator as cover. Once this classification is performed by the discriminator, the *loss* is computed. Then both networks, generator and discriminator, are concatenated and the loss propagates through the combined network as a standard CNN using the back-propagation algorithm and updating the network weights. However, discriminator weights are frozen so only the generator weights are updated. (see Fig. 2).

4. Evolving GAN architectures for steganography

The second phase of this work consists on designing an algorithm based on evolutionary computation to find the best GAN architecture. There are some articles in the literature which take a similar approach, such as *Evodeep* (Martín, Fuentes-Hurtado, Naranjo, & Camacho, 2017; Martín, Lara-Cabrera, Fuentes-Hurtado, Naranjo, & Camacho, 2018), which implements a genetic algorithm to find the best architecture and hyper-parameters for CNNs. It evolves a population of networks modifying the number and hyper-parameter of layers in each individual. In this research, the designed method is a genetic algorithm which evolves

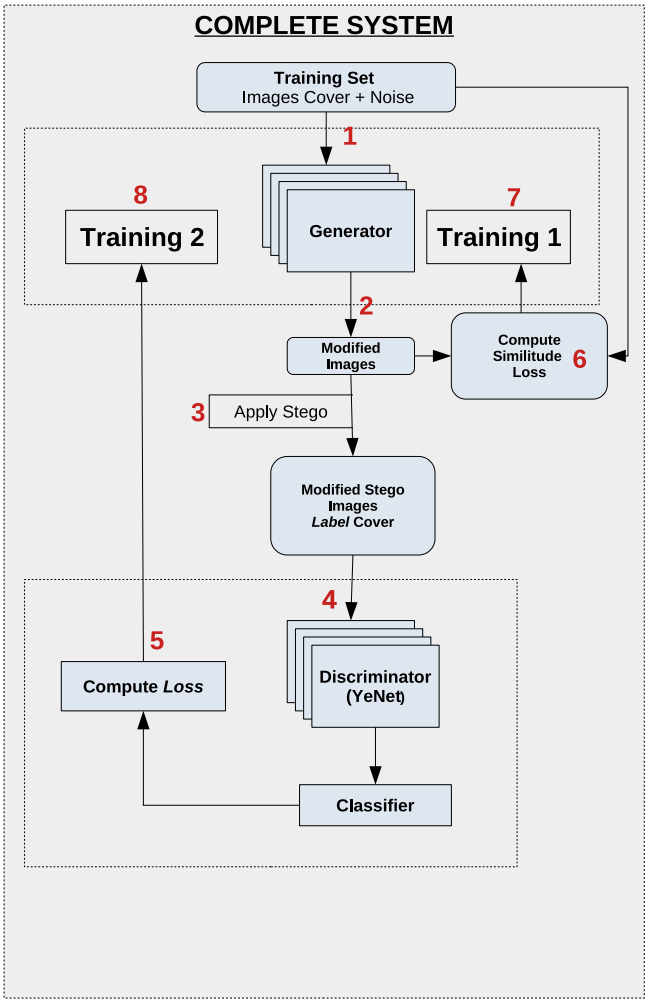

Fig. 3. Diagram describing the GAN model. The model steps are indicated by the numbers in the image. Images goes through 2 different ways: the path 1-2-6-7, which updates the generator using the cosine similarity loss; and the path 1-2-3-4-5-8, which updates the generator based on the discriminator loss. Steps 1–2 are shared and only performed once. The steps are: (1) the generator input is a 2 channel image consisting of a cover image and Gaussian noise, (2) the generator modifies the input image, (3) the steganographic module that embeds the steganographic message and label them as cover, (4) the stego images labelled as cover are classified by the discriminator, (5) the discriminator loss is computed using Adam optimiser, (6) the similarity loss is computed using the cosine similarity function between the modified and original images, (7) the generator weights are updated using the similarity loss, (8) the generator weights are updated using the discriminator loss.

a population of generators. Coral Reef-based optimisation has also be used to optimise the parameters of CNNs (Martín, Lara-Cabrera et al., 2020; Martín, Vargas, Gutiérrez, Camacho and Hervás-Martínez, 2020).

In this section, the different components and operators of the proposed genetic algorithm are proposed, defining the population codification, the crossover and mutation operators and the evaluation strategy.

4.1. Codification

The codification defines the architecture that can be generated by the algorithm. Its main goal is to find the architectures that best fit the problem (modify images to deceive the discriminator keeping the similarity to the original one as high as possible). However, the design of the population codification must be accomplished trying to keep the computational resources required as low as possible due to the computational complexity of the algorithm.

The codification will define CNNs with a similar structure to the generator proposed in the previous section. As shown in Fig. 4, the individuals are composed by 3 different groups. The first one consists on a group of transpose convolutional layers which enlarge the size of the image, then a group of convolutional layers that keep the same dimensions at the input and the output (called intermediate convolutions from now on), and finally a group of convolutional layers that decrease its size to its original size (named final convolutions in this work).

The generation of new individuals is guided by a very simple state machine (see Fig. 5). It is composed of only 2 states, the first one corresponds to the group of transpose convolutional layers and the second one to the intermediate group of convolutions. The final convolutional layers will be computed as a mirror of the transpose convolutional layers group, considering both the number of layers and their hyper-parameters.

Each state has a set of hyper-parameters that also evolve during the algorithm execution. Each one of the transpose convolutional layers (and the final convolutional layers) has the following hyper-parameters: *kernel size*, which can be 3 or 5; *stride*, which can be 1 or 2; the *activation function*, which can be *reLU* or *Leaky ReLU*; *number of filters*, which indicates the factor used to increase the number of filters by; and *batch normalisation*, which indicates if there is a batch normalisation layer after it. On the other side, the intermediate convolutional layers have the same hyper-parameters except for the *stride* which is always 1. The hyper-parameters of the final convolutional layers are the same that their corresponding mirror transpose convolutional layer.

In addition, the hyper-parameters are limited to prevent the individuals to growth in excess. The stride can only be 2 in one of the transpose convolutional layers, setting it to 1 in all the layers posterior to the first one with stride 2. This will prevent the individuals to achieve too high dimensions. The number of filters can be 1, 2 or 3, and the number of filters of the layer is computed multiplying the previous layer number of filters by this factor. However, it cannot exceed a certain value defined as a parameter of the genetic algorithm. All the layers that would exceed this limit will be set to it. Finally, the last layer of the network always has a single filter and its activation function is the *tanh* function, as its recommended in the literature (Radford et al., 2015).

4.2. Operators

The genetic algorithm has 2 operators: the crossover operator, which generates 2 descendants from 2 parents; and the mutation operator, which can modify the generated descendants.

4.2.1. Crossover

The crossover operator combines 2 individuals to generate 2 new descendants. It is composed of 2 different stages. The first one is the external crossover, which performs a 1-point crossover between the layers of both parents. First, the number of layers of the new descendant is selected from the range of possible values (set as genetic algorithm parameter). Then the first point is randomly selected from the first parent from those who allow to generate valid individuals, i.e, those who allow to generate an individual from the selected number of layers. Then the cross point in the second parent is generated from those who generate validate individuals, i.e, from those which keep the selected number of layers and from the same group than in the first parent (transpose convolution or intermediate convolution).

The second stage is the internal crossover. It performs an uniform crossover between the hyper-parameters of all layers. The individual codification for which the final convolutions are computed from the group of transpose convolutions allow to generate always valid individuals.

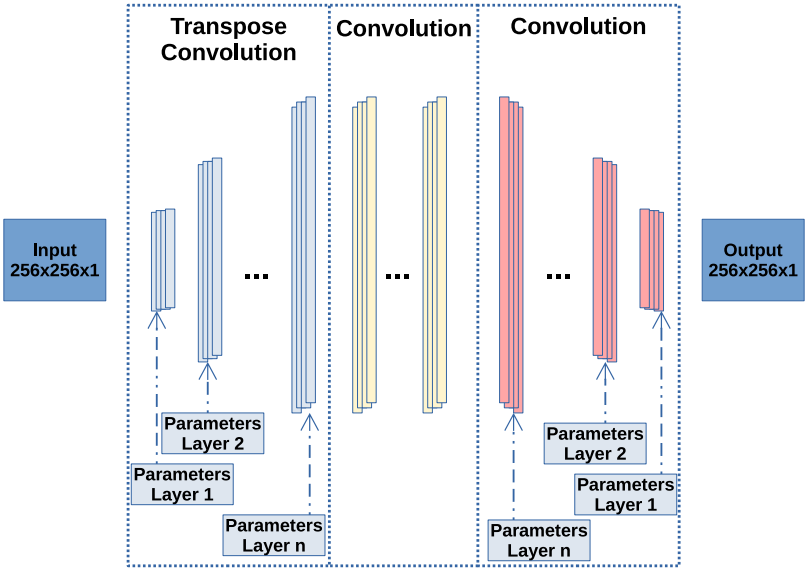

Fig. 4. Diagram that describes the architectures that the individuals can codify. It is composed by 3 group of layers: the transpose convolutional layers (left), the intermediate convolutional layers (middle) and the final convolutional layers (right).

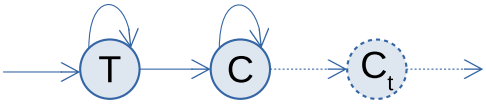

Fig. 5. State machine that rules the individuals generation. It is composed by 2 states: transpose convolutional layers (T) and intermediate convolutional layers (C); plus the mirror layers (C_t).

4.2.2. Mutation

The mutation operator is performed after crossover and it is also composed of 2 stages. In the first one there is a probability called *new layer probability*, which is set as parameter of the genetic algorithm, of adding a new layer in a random location of the network. The location where the layer is added is selected from those which do not increase the individual size above the layer limit. It must be considered that if a transpose convolutional layer is added, then a convolutional layer is added in the group of final convolutions (the mirror layer).

The second stage consists on a random mutation of each one of the hyper-parameters of each one of the layers of the network. The mutation probability is set as a hyper-parameter of the genetic algorithm, and it is called from now on *mutation probability*.

4.3. Evaluation

The last component of the genetic algorithm is the individual evaluation. To evaluate an individual, a GAN where the generators corresponds to its codification must be implemented. Then, the GAN model is trained during a predefined number of epochs, set as a parameter of the genetic algorithm. The GAN is trained using the training set, and evaluated with the validation set. The discriminator loss and similarity loss obtained with the validation set are used as fitness value of the individual, which allow to select the best individuals between the old and new population.

4.4. Selection

The selection algorithm must choose between the individuals of the current population and their descendants. According to its criterion, a number of individuals equal to the population size must be selected. The selection algorithm used in this work is called NSGA-II or *Non dominated sorting genetic algorithm* (Deb, Pratap, Agarwal, & Meyarivan, 2002). This algorithm selects the individuals of the Pareto Front and choose between them using a *crowding* function, i.e, it penalises closer solutions. If there are not enough solutions in this first selection, the process is repeated with elements that do not belong to the First Pareto Front.

5. Experiments

A set of experiments was run in order to assess the performance of the method proposed. With this purpose, 3 different experiments have been carried out using a well known steganography dataset. The first one measures the manually designed GAN model capacity to generate images which can deceive the discriminator. The second experiment is focused on the analysis of the quality of the genetic algorithm and compares the solutions obtained with the GAN designed in the previous experiment. Finally, in the third experiment, the discriminator trained again with the images generated with the genetic GAN model.

5.1. Dataset

In this work the *BOSSBase v1.0.1* dataset (Bas, Filler, & Pevnỳ, 2011) was used. It is composed of 10 000 grey and white images, i.e, each one is formed by a single canal with pixel values between 0 and 255 and image size of 256 × 256 pixels. These images represent the *cover* images. Then, the LSBr algorithm of the library *Aletheia* was used to create a *stego* version for each of the cover images, generating a dataset of 10 000 cover and 10 000 stego images. (see Fig. 6).

Once the dataset has been generated, it was divided into random training, validation and test sets. Each of these sets is formed by a set of cover images and their related stego images. The training, validation and testing set include 50%, 10% and 40% of the images respectively.

5.2. Experiment 1: stand-alone GAN

In order to evaluate the stand-alone GAN model, several tests were performed with the architecture shown in Section 3. The discriminator has been pre-trained using the *BOSSBasev1.0.1* dataset, and the results achieved can be seen in Table 2. The training process was executed for 40 epochs.

The pre-trained YeNet model is used as discriminator in the GAN model, which is trained during another 40 epochs, using an *Adam*

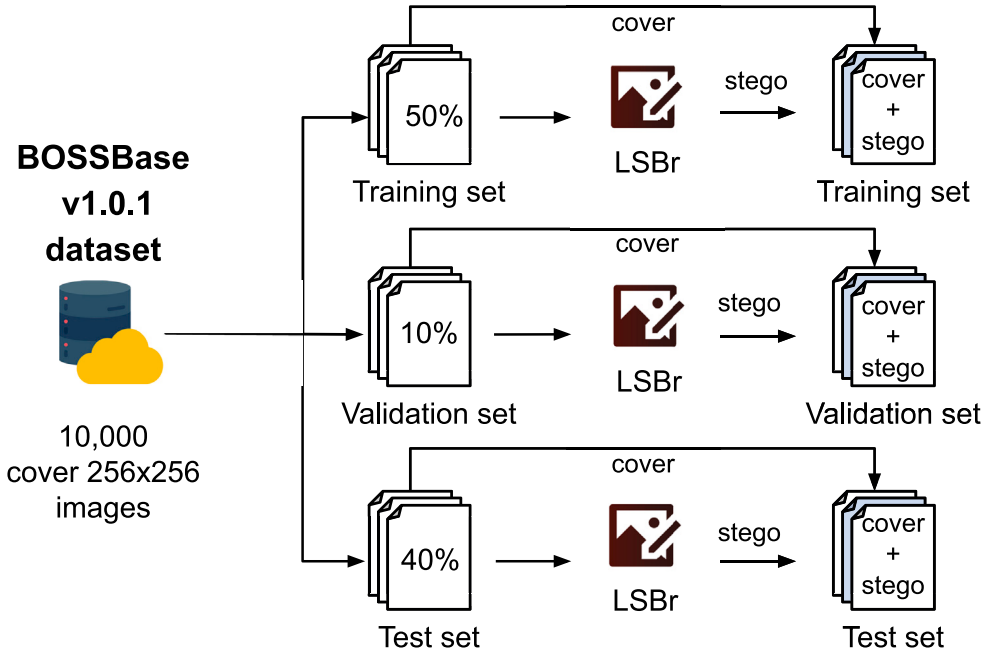

Fig. 6. Partitions created in the BOSSBase v1.0.1 dataset. For each of the partitions, training, validation and test, the LSBr algorithm is used to create the stego version of each image. Thus each partition contains a set of cover images together with their respective stego version.

Table 2
Accuracy results of the discriminator (YeNet model Ye et al. (2017)) pre-training.

	Accuracy	True positive rate	True negative rate
Training set	0.967	0.962	0.972
Validation set	0.932	0.936	0.928
Test set	0.918	0.916	0.92

optimiser with *learning rate* 0.0002 and *beta1* 0.9. For every epoch 2 measures are obtained, the similitude mean between the generated images and the training set, and the *discriminator loss*, which is the loss obtained with the discriminator using only the *fake* or *generated* images with the generator and with a steganaographic message embedded.

These 2 measures will be used to train the model in a multi-objective scenario, including to maximise similitude and the miss-classifications generated by the discriminator of *fake* images. The goal of this approach is to find the best weights for the generator with the goal of producing miss-classifications in the steganalysis model (discriminator). For this purpose and to choose the best result, the mean similitude between the generated images and the original ones and the loss value obtained when classifying fake stego images with the discriminator. At the end of each epoch, these 2 measures are computed using the validation set to guide the training process.

At the end of the training, both metrics are calculated for all the models generated, which form a Pareto Front. In Table 3, one can see the 10 solutions that belong to this Pareto Front, i.e, those which are not improved in both measures by any other solution. The *discriminator* loss achieves its best results in the fourth epoch, minimising loss and accuracy. However, the *similitude* loss decreases in later training epochs. Both measures of every solution keep low values during the complete training process, being the miss-classifications of fake images higher than 99% in most of the cases. The *similitude* loss, although it decrease in the last epochs, it keeps acceptable values during the whole training.

The chosen model is the one trained during 35 epochs, which is the one showing the best balance the *similitude* loss and the *discriminator* loss. In Table 4 it is possible to observe the results of this model in the test set.

Table 3
Solutions obtained with the stand-alone GAN located at the Pareto Front. Values indicate performance in the validation set.

	Accuracy	Discriminator loss	Similitude loss	Epoch
1	0.007398	**0.00548**	0.01	4
2	0.002	0.0138	0.003	13
3	0.002	0.0137	0.0037	19
4	0.001	0.0056	0.00417	26
5	0.003	0.0139	0.00186	29
6	0.007	0.024	0.0018	31
7	0.007	0.0252	0.00171	32
8	0.008	0.0296	0.00169	35
9	0.011	0.0344	**0.0015**	36
10	0.009	0.0298	0.00166	38

Table 4
Final GAN evaluated using the test set.

Accuracy	Discriminator loss	Similitude loss
0.008	0.03075876	0.00167084

The similitude loss is very close to the best result of the Pareto Front, and the miss-classifications are higher than 99% with stego fake images. Thus, it is possible to confirm that the final model achieves the desired goals. As shown in Fig. 7, the visual difference between the original image and the same one after being modified are indistinguishable. (see Fig. 8).

5.3. Experiment 2: Evolutionary GAN

As explained in Section 4, the goal of the genetic GAN is to find the best generator architecture and hyper-parameters. The genetic algorithm will be executed 5 times to assess its performance. The training set has been employed to train each individual of the population, and the validation set has been used to evaluate each individual and obtain the fitness values (similitude loss and discriminator loss).

The parameters used for the algorithm execution are the following: *population size* (μ), which represents the number of generators that

Fig. 7. Original and modified stego images. (Left) Original image with secret message embedded. (Right) Same image modified with the GAN's generator.

compose the population at each generation; *lambda* (λ), which is the number of crossovers (and mutations) performed at each epoch, so at each generation $2 \cdot \lambda$ individuals are generated; *crossover probability*, *mutation probability* and *new layer probability*, which regulates the descendant generation as explained in Section 4.2; *max depth*, which is the maximum number of kernels that any generator layer can possess; and *epochs per individual*, which is the number of epochs during each generator is evaluated. Each execution of the genetic algorithm last 10 generations. The specific values are described in Table 5.

To evaluate the individuals performance, both the discriminator loss and the similitude between original and modified images were considered, but also the execution time. The genetic algorithm's execution time is the highest disadvantage it has, so smaller generators are favoured.

Table 6 shows the results in the test set of the best individual of each one of the 5 executions. The results are comparable to those obtained with the manually designed generator, but they are more efficient, given that they were train over 2 epochs.

On the other side, Fig. 11 displays the evolution in terms of similitude loss and accuracy of the best individual at each generation. As can be observed, it is only required 3 generations to find a generator

Table 5
Hyper-parameters employed during the genetic algorithm execution.

Hyperparameter	Value	Hyperparameter	Value
Generations	10	Max depth	32
μ	10	λ	5
New layer probability	0.3	Crossover probability	0.5
Mutation probability	0.25	Epochs per individual	2

Table 6
Performance of the best individual obtained in each execution of the genetic algorithm.

Execution	Stego accuracy	Discriminator loss	Similitude	Training time (s)
1	0.0005	0.0084	0.0032	191
2	0.01	0.022	0.004	191
3	0.0001	0.00867	0.0032	123
4	0.00075	0.0101	0.0032	78
5	0.00275	0.0106	0.0032	68

with a similitude and accuracy close to the minimum. In addition, the similitude loss presents small variations during the whole training.

Finally, the best individual of each execution has been trained under the same conditions, described in Section 5.2, i.e, during 40 epochs using only the training set. Results are provided in Figs. 9 and 10.

The similitude loss has the same distribution than the generator designed in Experiment 1. The main difference lies in the evolution of loss in the discriminator which, although in 3 of the five executions achieves quasi-optimal values in the first 5 epochs, there are 2 executions (3 and 5) that take around 15 epochs. This is due to the weights initialisation, a fact highly correlated to the time required to find the best possible solutions.

5.4. Experiment 3: Re-training discriminator

The last experiment performed in this work consists on retraining the discriminator with the results obtained with the generator. The goal is to take advantage of the fact that the best individuals achieve high quality results in a small number of epochs, and that different generators, with similar performance measures, modify images in a different way each.

For this reason, a new training strategy was proposed, considering 2 new parameters called θ_0 and θ_1, which control the quality of each new generated individual. At the end of each individual evaluation, its performance is measured comparing the accuracy of the discriminator using the generated stego images (from the validation set) with the θ_0 value, which has been set to 0.5. Thus, if more than 50% of the fake stego images are miss-classified by the discriminator, then the individual is qualified to re-train the discriminator. The θ_1 parameter

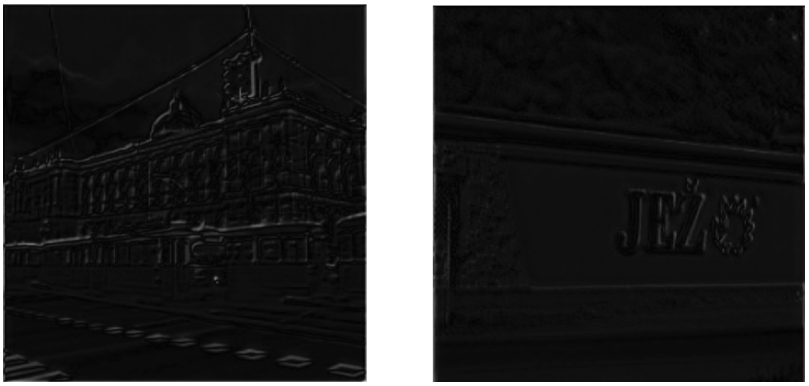

Fig. 8. Difference between the original image and the image including stenography. Lighter pixels show where changes have been made. As can be seen, the differences are minimal, practically impossible for the human eye to distinguish.

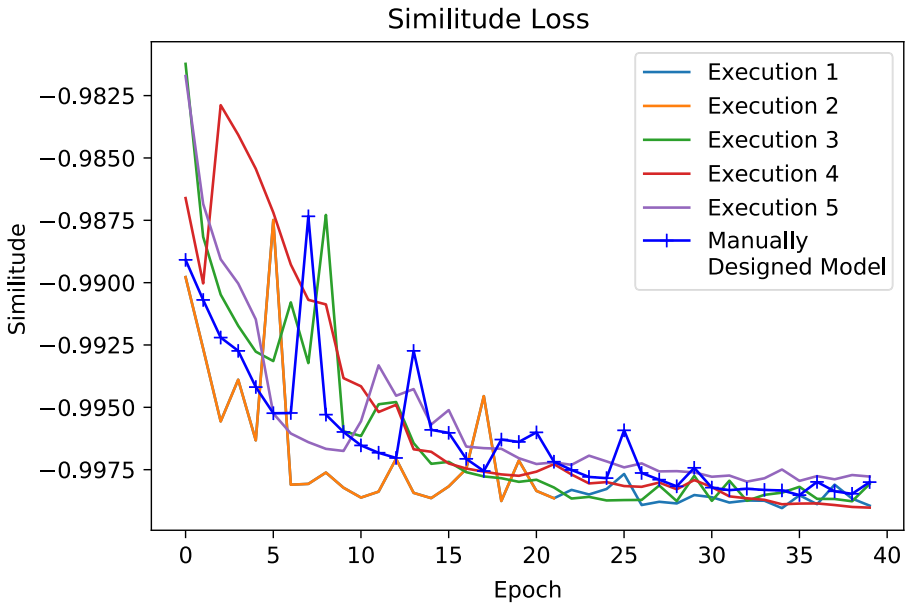

Fig. 9. Evaluation of the similitude loss of the best individual across executions.

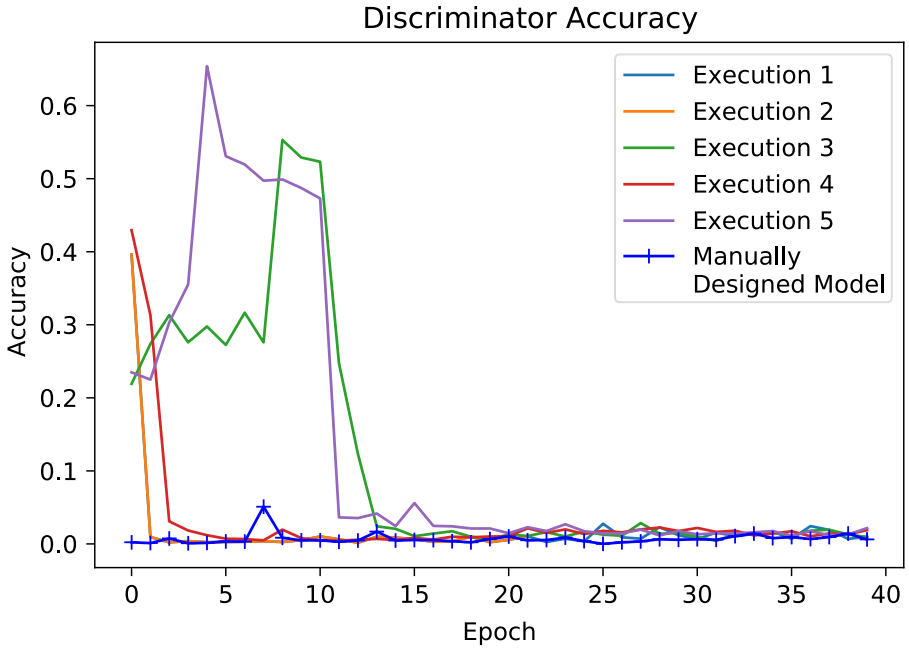

Fig. 10. Evaluation of the Fake Stego Images accuracy using the discriminator for the best individual across executions.

has a similar goal, but focused on the similarity loss. If both conditions are met, a new fake stego image is generated and stored for each cover sample in the training set.

Finally, at the end of each generation, the discriminator is retrained during a fixed number of epochs (also set as a hyper-parameter). This hyper-parameter was set to 5. For the training, both the original training set and the images generated during the individuals evaluation were evaluated. The performance of the re-trained discriminator is measured using the test set and the same metrics than in previous experiments. The population of each generation is evaluated using the discriminator retrained in the previous one.

Table 7 shows the results of the ten executions of the genetic algorithm retraining the discriminator and the results of the pre-trained YeNet model. In all the executions the final discriminator classifies the original test set better than the discriminator at the experiment beginning.

Finally, considering the results during the execution (Fig. 12), they are very unstable. This fact evidences how the algorithm pursues a balance between cover and stego accuracy, which causes high drops at some points. This is due to the over-fitting of one class, causing a high number of miss-classifications in the other.

6. Discussion

The previous experiments have demonstrated how the GAN architecture proposed in this work allows to deceive a steganalysis model. This involves a population of generators that evolve aiming to maximise the error rate of a steganalysis model, implemented by the discriminator, while minimising the number of changes required to embed

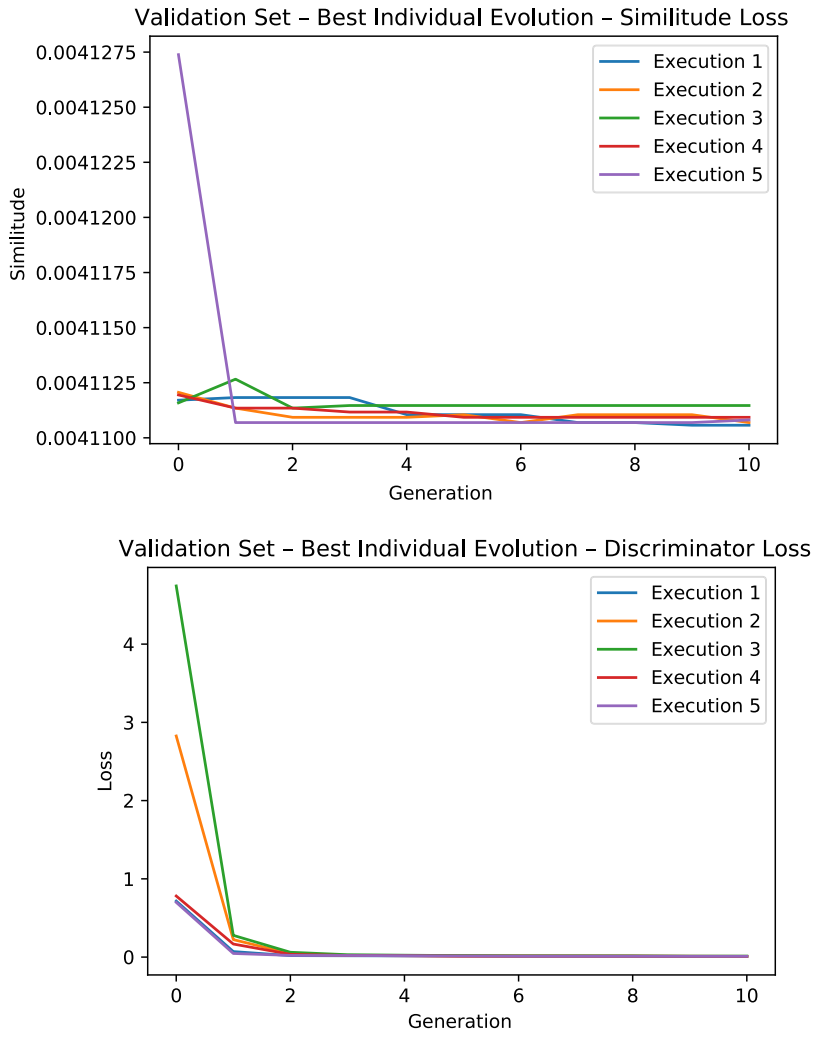

Fig. 11. The top figure displays the best Similitude Loss value at the end of each generation. The bottom plot shows the best Discriminator Loss value at the end of each generation.

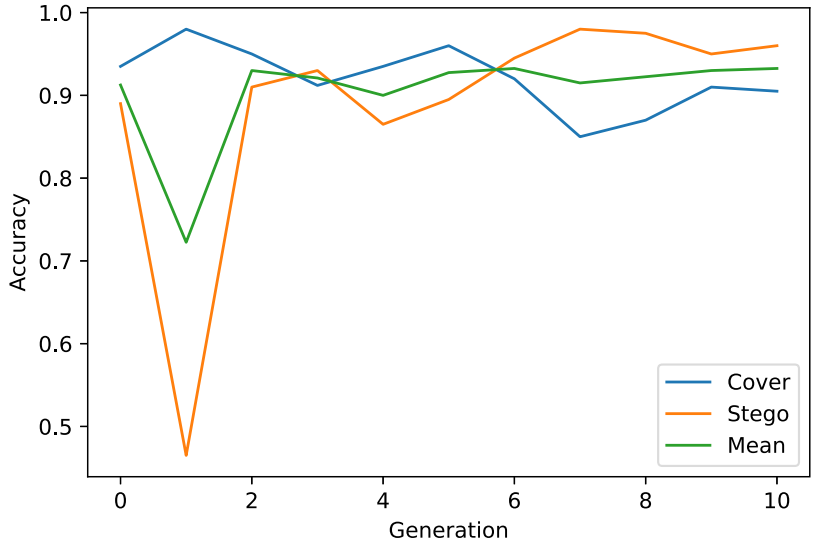

Fig. 12. Performance measurement using the test set of the discriminator during the genetic algorithm execution.

Table 7
Best results for the re-trained discriminator measured using the test set.

Execution	Global accuracy	Cover accuracy	Stego accuracy
Pre-Trained YeNet	0.918	0.92	0.916
1	0.923750	0.89675	0.95075
2	0.928750	0.94700	0.91050
3	0.931375	0.91350	0.94925
4	0.928375	0.93875	0.91800
5	0.929625	0.91750	0.94175
6	0.925125	0.91675	0.93350
7	0.931125	0.90275	0.95950
8	0.930875	0.93050	0.93125
9	0.923250	0.93725	0.90925
10	0.925000	0.93450	0.91550

the secret message into the cover image. The evaluation of the GAN architecture in Experiment 1 has evidenced that this architecture is able to successfully reduce the ability of the discriminator to distinguish between cover and stego images, decreasing the accuracy from 90% to 10%.

The second experiment showed how a genetic algorithm can be used to generate valid composition of layers to build a generator. With a small number of epochs, the algorithm is able to produce individuals which define a sequence of layers that allows to apply a series of changes to prepare the image to later embed a messages. The third experiment has demonstrated that the combination of the stand-alone GAN with the genetic algorithm can be used to successfully deceive the discriminator while introducing a high number of variations, which hampers the capacity of the discriminator to detect manipulated images.

To the best of our knowledge, no other research has leveraged the benefits of a GAN architecture in conjunction with a genetic algorithm to prepare images before the message is embedded into the cover medium. As showed throughout the experiments, the approach is compared against a state-of-the-art CNN architecture employed for image steganalysis. This architecture, the YeNet model (Ye et al., 2017), has been trained over the same dataset for a fair comparison. While the YeNet model is able to detect the presence of steganographic techniques with 91.8% accuracy in the original dataset, the application of the GAN and the best individual provided by the genetic algorithm allows to reduce this number to 0.8%. This is a large reduction which evidences the limitations of standard CNN-based architectures for image steganalysis, while proving that GAN schemes are successful at building stego images undetectable.

Other researchers have proposed the use of genetic algorithms to define a strategy to embed messages into images (Kanan & Nazeri, 2014a) or to find the most adequate pixels to embed the message (Tseng et al., 2008). However, these approaches do not consider the use of Convolutional Neural Networks used as steganalysis methods, which can be continuously trained to improve. The combination of genetic algorithms and steganography has been studied in the literature (Mandal, Mukherjee, Paul, & Chatterji, 2022), and researches have proposed different approaches with different goals. Notwithstanding, the use of an evolutionary strategy to prepare an image by applying modifications in the spatial domains is a novel approach proposed in this work.

Similar works have studied the use of Generative Adversarial Networks to learn change probabilities for each pixel in the spatial domain of the cover image, so later different distortions are introduced but minimising the detection probability (Tang et al., 2017). Similarly, again without introducing an evolutionary strategy, authors have employed a GAN to improve the quality of images embeddings hidden messages (Qin et al., 2020). With the goal of embedding images as payload, researchers have also evidenced that encoder–decoder architectures are a powerful instrument, proposing a new loss function (ur Rehman, Rahim, Nadeem, & ul Hussain, 2019). This research, in contrast, focuses on the image space before applying the steganography method. In this line, a state-of-the-art research proposes a new approach for embeddings the hidden message layer by layer, seeking for a better integration of the cover image. The authors employ a encoding and a decoding networks, which is the steganography method used (Gan & Zhong, 2021).

In case of the application of evolutionary strategies, state-of-the-art literature has mostly focused on the use of genetic algorithms. One typical combination are methods based on the integer wavelet transform, where the genetic algorithm is used to select coefficients (Sabeti et al., 2022), while other authors employ the genetic algorithm to generate stego images based on different functions (Raja et al., 2007) and other researchers are focused on the frequency domain (Miri & Faez, 2017).

With a different objective, genetic algorithms have been used in the steganography domain to generate images with characteristics that can be the key to break the inspection of steganalysis systems (Wu & Shih, 2006). An evolutionary strategy can also be used to improve the quality of the stego image, through a genetic algorithm and a Optimal Pixel Adjustment Process (Tseng et al., 2008). Finally, other researchers have also studied the use of genetic algorithms to encrypt the secret message and later use the LSB algorithm (Khodaei & Faez, 2010) or a method based on modelling steganography as a search and optimisation problem (Kanan & Nazeri, 2014b).

7. Conclusions and future work

In summary, the scheme proposed in this research involves a GAN and a genetic algorithm, in charge of defining the best architecture for the generator. In the experimentation section, three different experiments have allowed (1) to measure the performance of the GAN architecture, (2) to evaluate the implementation of the genetic algorithm an its performance at finding valid sequences of layers to build a generator of cover images adapted for steganography and (3) to evaluate if the combination of the GAN architecture proposed and the genetic algorithm is successfully at deceiving a steganalysis model.

The approach has proven to be able to improve the generator performance by generating new adapted cover images where a hidden message can be safely introduced. In comparison to previous research, this agnostic approach aims at providing an instrument to prepare a cover medium to later introduce the message. Thus, different steganography algorithms can be used, including combinations with cryptography algorithms to avoid accessing the message if the medium is intercepted. Moreover, the use of an evolutionary strategy allows to continuously build new generators which introduce different changes, thereby hampering the application of classic steganalysis models.

Future work involves analysing other steganography algorithms, including those centred on the spatial domain but also in the frequency domain. Furthermore, other evolutionary strategies can also be analysed, evaluating how different schemes can be lead to different results. Other architectures can also be tested. Lighter architectures would allow to train the GAN model more epochs during the individuals evaluation. It would be also interesting to analyse if the generator architecture can be generalised (a good architecture is good against any steganalysis model) or if the advantages of the architecture depends on the discriminator employed. With this purpose the discriminator could be replaced by other steganalysis CNN model of the state of the art. Besides, the approach proposed could be also extended to target other types of steganography and steganalysis techniques and also other types of cover mediums, such as audio or video.

CRediT authorship contribution statement

Alejandro Martín: Conceptualization, Data curation, Writing, Funding acquisition, Supervision. **Alfonso Hernández:** Conceptualization, Software, Validation, Visualization, Methodology. **Moutaz Alazab:** Conceptualization, Software, Visualization, Writing, Methodology. **Jason Jung:** Conceptualization, Software, Validation, Writing, Data curation. **David Camacho:** Conceptualization, Writing, Resources, Funding acquisition, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: David Camacho reports financial support was provided by Community of Madrid. David Camacho reports financial support was provided by Spanish Ministry of Science and Innovation.

Data availability

Data will be made available on request.

Acknowledgements

This research has been supported by Comunidad Autónoma de Madrid, Spain under S2018/TCS-4566 (CYNAMON) grant, by grant PID2020-117263GB-100 funded by MCIN/AEI/10.13039/5011000 11033 and, as appropriate, by "ERDF A way of making Europe", by the "European Union" or by the "European Union NextGenerationEU/PRTR" and by Comunidad Autónoma de Madrid under: "Convenio Plurianual with the Universidad Politécnica de Madrid in the actuation line of *Programa de Excelencia para el Profesorado Universitario*".

References

Abdalla, O. A. (2014). Optimizing the multilayer feed-forward artificial neural networks architecture and training parameters using genetic algorithm. *International Journal of Computer Applications*, *96*(10), 42–48. http://dx.doi.org/10.5120/16832-6596.

Amirtharajan, R., & Rayappan, J. B. B. (2013). Steganography-time to time: A review. *Research Journal of the Information Technology*, *5*, 53–66. http://dx.doi.org/10.3923/rjit.2013.53.66.

Bas, P., Filler, T., & Pevnỳ, T. (2011). Break our steganographic system: the ins and outs of organizing BOSS. In *International workshop on information hiding* (pp. 59–70). Springer, http://dx.doi.org/10.1007/978-3-642-24178-9_5.

Cheddad, A., Condell, J., Curran, K., & Mc Kevitt, P. (2010). Digital image steganography: Survey and analysis of current methods. *Signal Processing*, *90*(3), 727–752. http://dx.doi.org/10.1016/j.sigpro.2009.08.010.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in Neural Information Processing Systems*, *29*, URL https://proceedings.neurips.cc/paper/2016/file/7c9d0b1f96aebd7b5eca8c3edaa19ebb-Paper.pdf.

Chen, X., Pan, J., Jiang, K., Li, Y., Huang, Y., Kong, C., et al. (2022). Unpaired deep image deraining using dual contrastive learning. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)* (pp. 2007–2016). http://dx.doi.org/10.1109/CVPR52688.2022.00206.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. http://dx.doi.org/10.1109/4235.996017.

Dumitrescu, S., Wu, X., & Wang, Z. (2002). Detection of LSB steganography via sample pair analysis. In *International workshop on information hiding* (pp. 355–372). Springer, http://dx.doi.org/10.1007/3-540-36415-3_23.

El-Khalil, R., & Keromytis, A. D. (2004). Hydan: Hiding information in program binaries. In J. Lopez, S. Qing, & E. Okamoto (Eds.), *Information and communications security* (pp. 187–199). Berlin, Heidelberg: Springer Berlin Heidelberg.

Fard, A. M., Akbarzadeh-T, M.-R., & Varasteh-A, F. (2006). A new genetic algorithm approach for secure JPEG steganography. In *2006 IEEE international conference on engineering of intelligent systems* (pp. 1–6). IEEE, http://dx.doi.org/10.1109/ICEIS.2006.1703168.

Finlayson, S. G., Bowers, J. D., Ito, J., Zittrain, J. L., Beam, A. L., & Kohane, I. S. (2019). Adversarial attacks on medical machine learning. *Science*, *363*(6433), 1287–1289. http://dx.doi.org/10.1126/science.aaw4399.

Fridrich, J., & Kodovsky, J. (2012). Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, *7*(3), 868–882. http://dx.doi.org/10.1109/TIFS.2012.2190402.

Gan, Z., & Zhong, Y. (2021). A novel grayscale image steganography via generative adversarial network. In C. Xing, X. Fu, Y. Zhang, G. Zhang, & C. Borjigin (Eds.), *Web information systems and applications* (pp. 405–417). Cham: Springer International Publishing.

Gascón-Moreno, J., Salcedo-Sanz, S., Saavedra-Moreno, B., Carro-Calvo, L., & Portilla-Figueras, A. (2013). An evolutionary-based hyper-heuristic approach for optimal construction of group method of data handling networks. *Information Sciences*, *247*, 94–108. http://dx.doi.org/10.1016/j.ins.2013.06.017.

Geetha, S., & Kamaraj, N. (2010). Optimized image steganalysis through feature selection using MBEGA. http://dx.doi.org/10.48550/ARXIV.1008.2824, arXiv preprint arXiv:1008.2824.

Ghasemi, E., Shanbehzadeh, J., & Fassihi, N. (2010). High capacity image steganography using wavelet transform and genetic algorithm. In *World congress on engineering 2012. July 4-6, 2012. London, UK, Vol. 2188* (pp. 495–498). International Association of Engineers, URL http://www.iaeng.org/publication/IMECS2011/IMECS2011_pp495-498.pdf.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144. http://dx.doi.org/10.1145/3422622.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1904–1916. http://dx.doi.org/10.1109/TPAMI.2015.2389824.

Hu, D., Wang, L., Jiang, W., Zheng, S., & Li, B. (2018). A novel image steganography method via deep convolutional generative adversarial networks. *IEEE Access*, *6*, 38303–38314. http://dx.doi.org/10.1109/ACCESS.2018.2852771.

Huertas-Tato, J., Martín, A., Fierrez, J., & Camacho, D. (2022). Fusing CNNs and statistical indicators to improve image classification. *Information Fusion*, *79*, 174–187. http://dx.doi.org/10.1016/j.inffus.2021.09.012, URL https://www.sciencedirect.com/science/article/pii/S1566253521001883.

Hussain, M., Wahab, A. W. A., Idris, Y. I. B., Ho, A. T., & Jung, K.-H. (2018). Image steganography in spatial domain: A survey. *Signal Processing: Image Communication*, *65*, 46–66. http://dx.doi.org/10.1016/j.image.2018.03.012, URL https://www.sciencedirect.com/science/article/pii/S092359651830256X.

Kadhim, I. J., Premaratne, P., Vial, P. J., & Halloran, B. (2019). Comprehensive survey of image steganography: Techniques, evaluations, and trends in future research. *Neurocomputing*, *335*, 299–326. http://dx.doi.org/10.1016/j.neucom.2018.06.075.

Kanan, H. R., & Nazeri, B. (2014a). A novel image steganography scheme with high embedding capacity and tunable visual image quality based on a genetic algorithm. *Expert Systems with Applications*, *41*(14), 6123–6130. http://dx.doi.org/10.1016/j.eswa.2014.04.022.

Kanan, H. R., & Nazeri, B. (2014b). A novel image steganography scheme with high embedding capacity and tunable visual image quality based on a genetic algorithm. *Expert Systems with Applications*, *41*(14), 6123–6130. http://dx.doi.org/10.1016/j.eswa.2014.04.022, URL https://www.sciencedirect.com/science/article/pii/S095741741400219X.

Karampidis, K., Kavallieratou, E., & Papadourakis, G. (2018). A review of image steganalysis techniques for digital forensics. *Journal of Information Security and Applications*, *40*, 217–235. http://dx.doi.org/10.1016/j.jisa.2018.04.005.

Ke, Q., Ming, L. D., & Daxing, Z. (2018). Image steganalysis via multi-column convolutional neural network. In *2018 14th IEEE international conference on signal processing (ICSP)* (pp. 550–553). IEEE, http://dx.doi.org/10.1109/ICSP.2018.8652324.

Ker, A. D. (2005). Steganalysis of LSB matching in grayscale images. *IEEE Signal Processing Letters*, *12*(6), 441–444. http://dx.doi.org/10.1109/LSP.2005.847889.

Khodaei, M., & Faez, K. (2010). Image hiding by using genetic algorithm and LSB substitution. In A. Elmoataz, O. Lezoray, F. Nouboud, D. Mammass, & J. Meunier (Eds.), *Image and signal processing* (pp. 404–411). Berlin, Heidelberg: Springer Berlin Heidelberg, http://dx.doi.org/10.1007/978-3-642-13681-8_47.

Kodovsky, J., Fridrich, J., & Holub, V. (2011). Ensemble classifiers for steganalysis of digital media. *IEEE Transactions on Information Forensics and Security*, *7*(2), 432–444. http://dx.doi.org/10.1109/TIFS.2011.2175919.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, *25*, 1097–1105. http://dx.doi.org/10.1145/3065386.

Kumar, R., & Rattan, M. (2012). Analysis of various quality metrics for medical image processing. *International Journal of Advanced Research in Computer Science and Software Engineering*, *2*(11), 137–144.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., et al. (1990). Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems* (pp. 396–404). URL https://proceedings.neurips.cc/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf.

Lee, K., Westfeld, A., & Lee, S. (2007). Generalised category attack—improving histogram-based attack on jpeg LSB embedding. In *International workshop on information hiding* (pp. 378–391). Springer, http://dx.doi.org/10.1007/978-3-540-77370-2_25.

Leung, F. H.-F., Lam, H.-K., Ling, S.-H., & Tam, P. K.-S. (2003). Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural Networks*, *14*(1), 79–88. http://dx.doi.org/10.1109/TNN.2002.804317.

Li, L., Fan, M., & Liu, D. (2021). AdvSGAN: Adversarial image steganography with adversarial networks. *Multimedia Tools and Applications*, *80*(17), 25539–25555. http://dx.doi.org/10.1007/s11042-021-10904-1.

Lu, J.-c., Liu, F.-l., & Luo, X.-y. (2014). Selection of image features for steganalysis based on the Fisher criterion. *Digital Investigation*, *11*(1), 57–66. http://dx.doi.org/10.1016/j.diin.2013.12.001.

Mandal, P. C., Mukherjee, I., Paul, G., & Chatterji, B. (2022). Digital image steganography: A literature survey. *Information Sciences*, *609*, 1451–1488. http://dx.doi.org/10.1016/j.ins.2022.07.120, URL https://www.sciencedirect.com/science/article/pii/S002002552200809X.

Martín, A., Fuentes-Hurtado, F., Naranjo, V., & Camacho, D. (2017). Evolving deep neural networks architectures for android malware classification. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1659–1666). http://dx.doi.org/10.1109/CEC.2017.7969501.

Martín, A., Lara-Cabrera, R., Fuentes-Hurtado, F., Naranjo, V., & Camacho, D. (2018). EvoDeep: a new evolutionary approach for automatic deep neural networks parametrisation. *Journal of Parallel and Distributed Computing*, *117*, 180–191. http://dx.doi.org/10.1016/j.jpdc.2017.09.006.

Martín, A., Lara-Cabrera, R., Vargas, V. M., Gutiérrez, P. A., Hervás-Martínez, C., & Camacho, D. (2020). Statistically-driven Coral Reef metaheuristic for automatic hyperparameter setting and architecture design of convolutional neural networks. In *2020 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1–8). http://dx.doi.org/10.1109/CEC48606.2020.9185914.

Martín, A., Vargas, V. M., Gutiérrez, P. A., Camacho, D., & Hervás-Martínez, C. (2020). Optimising convolutional neural networks using a hybrid statistically-driven Coral Reef Optimisation algorithm. *Applied Soft Computing*, *90*, Article 106144. http://dx.doi.org/10.1016/j.asoc.2020.106144.

Marvel, L. M., Boncelet, C. G., & Retter, C. T. (1999). Spread spectrum image steganography. *IEEE Transactions on Image Processing*, *8*(8), 1075–1083. http://dx.doi.org/10.1109/83.777088.

Miri, A., & Faez, K. (2017). Adaptive image steganography based on transform domain via genetic algorithm. *Optik*, *145*, 158–168. http://dx.doi.org/10.1016/j.ijleo.2017.07.043.

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. http://dx.doi.org/10.48550/ARXIV.1411.1784, URL https://arxiv.org/abs/1411.1784.

Naito, H., & Zhao, Q. (2019). A new steganography method based on generative adversarial networks. In *2019 IEEE 10th international conference on awareness science and technology (ICAST)* (pp. 1–6). http://dx.doi.org/10.1109/ICAwST.2019.8923579.

Neeta, D., Snehal, K., & Jacobs, D. (2006). Implementation of LSB steganography and its evaluation for various bits. In *2006 1st international conference on digital information management* (pp. 173–178). IEEE, http://dx.doi.org/10.1109/ICDIM.2007.369349.

Paulson, L. (2006). New system fights steganography, News Briefs. *IEEE Computer Society*, *39*(8), 25–27. http://dx.doi.org/10.1109/MC.2006.273.

Pevny, T., Bas, P., & Fridrich, J. (2010). Steganalysis by subtractive pixel adjacency matrix. *IEEE Transactions on Information Forensics and Security*, *5*(2), 215–224. http://dx.doi.org/10.1145/1597817.1597831.

Qian, Y., Dong, J., Wang, W., & Tan, T. (2015). Deep learning for steganalysis via convolutional neural networks. In *Media watermarking, security, and forensics 2015, Vol. 9409*. International Society for Optics and Photonics, Article 94090J.

Qin, J., Wang, J., Tan, Y., Huang, H., Xiang, X., & He, Z. (2020). Coverless image steganography based on generative adversarial network. *Mathematics*, *8*(9), http://dx.doi.org/10.3390/math8091394, URL https://www.mdpi.com/2227-7390/8/9/1394.

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. http://dx.doi.org/10.48550/ARXIV.1511.06434, URL https://arxiv.org/abs/1511.06434.

Raja, K. B., Kumar K., K., Kumar N., S., Lakshmi, M. S., Preeti, H., Venugopal, K. R., et al. (2007). Genetic algorithm based steganography using wavelets. In P. McDaniel, S. K. Gupta (Eds.), *Information systems security* (pp. 51–63). Berlin, Heidelberg: Springer Berlin Heidelberg, http://dx.doi.org/10.1007/978-3-540-77086-2_5.

Ramezani, M., & Ghaemmaghami, S. (2010). Towards genetic feature selection in image steganalysis. In *2010 7th IEEE consumer communications and networking conference* (pp. 1–4). IEEE, http://dx.doi.org/10.1109/CCNC.2010.5421805.

ur Rehman, A., Rahim, R., Nadeem, S., & ul Hussain, S. (2019). End-to-end trained CNN encoder-decoder networks for image steganography. In L. Leal-Taixé, & S. Roth (Eds.), *Computer vision – ECCV 2018 workshops* (pp. 723–729). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-11018-5_64.

Sabeti, V., Sobhani, M., & Hasheminejad, S. M. H. (2022). An adaptive image steganography method based on integer wavelet transform using genetic algorithm. *Computers & Electrical Engineering*, *99*, Article 107809. http://dx.doi.org/10.1016/j.compeleceng.2022.107809, URL https://www.sciencedirect.com/science/article/pii/S0045790622001094.

Shi, H., Dong, J., Wang, W., Qian, Y., & Zhang, X. (2017). SSGAN: secure steganography based on generative adversarial networks. In *Pacific Rim conference on multimedia* (pp. 534–544). Springer, http://dx.doi.org/10.1007/978-3-319-77380-3_51.

Stanković, R. S., & Falkowski, B. J. (2003). The haar wavelet transform: its status and achievements. *Computers & Electrical Engineering*, *29*(1), 25–44. http://dx.doi.org/10.1016/S0045-7906(01)00011-8, URL https://www.sciencedirect.com/science/article/pii/S0045790601000118.

Tang, W., Tan, S., Li, B., & Huang, J. (2017). Automatic steganographic distortion learning using a generative adversarial network. *IEEE Signal Processing Letters*, *24*(10), 1547–1551. http://dx.doi.org/10.1109/LSP.2017.2745572.

Tseng, L.-Y., Chan, Y.-K., Ho, Y.-A., & Chu, Y.-P. (2008). Image hiding with an improved genetic algorithm and an optimal pixel adjustment process. In *2008 eighth international conference on intelligent systems design and applications, Vol. 3* (pp. 320–325). IEEE, http://dx.doi.org/10.1109/ISDA.2008.235.

Visavalia, S. R., & Ganatra, A. (2014). Improving blind image steganalysis using genetic algorithm and fusion technique. *Journal of Computer Science*, *1*, 40–46, URL https://www.iosrjournals.org/iosr-jce/papers/ICAET-2014/volume-1/7.pdf?id=7557.

Volkhonskiy, D., Nazarov, I., & Burnaev, E. (2020). Steganographic generative adversarial networks. In W. Osten, & D. P. Nikolaev (Eds.), *Twelfth international conference on machine vision (ICMV 2019), Vol. 11433* (p. 114333M). SPIE, International Society for Optics and Photonics, http://dx.doi.org/10.1117/12.2559429.

Wang, Z., & Bovik, A. C. (2002). A universal image quality index. *IEEE Signal Processing Letters*, *9*(3), 81–84. http://dx.doi.org/10.1109/97.995823.

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

Wang, Z., Gao, N., Wang, X., Qu, X., & Li, L. (2018). SSteGAN: self-learning steganography based on generative adversarial networks. In *International conference on neural information processing* (pp. 253–264). Springer, http://dx.doi.org/10.1007/978-3-030-04179-3_22.

Watson, A. B., et al. (1994). Image compression using the discrete cosine transform. *Mathematica Journal*, *4*(1), 81, URL https://humansystems.arc.nasa.gov/publications/mathjournal94.pdf.

Westfeld, A., & Pfitzmann, A. (1999). Attacks on steganographic systems. In *International workshop on information hiding* (pp. 61–76). Springer, http://dx.doi.org/10.1007/10719724_5.

Wu, Y.-T., & Shih, F. (2006). Genetic algorithm based methodology for breaking the steganalytic systems. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *36*(1), 24–31. http://dx.doi.org/10.1109/TSMCB.2005.852474.

Wu, D.-C., & Tsai, W.-H. (2003). A steganographic method for images by pixel-value differencing. *Pattern Recognition Letters*, *24*(9–10), 1613–1626. http://dx.doi.org/10.1016/S0167-8655(02)00402-6.

Wu, H.-C., Wu, N.-I., Tsai, C.-S., & Hwang, M.-S. (2005). Image steganographic scheme based on pixel-value differencing and LSB replacement methods. *IEE Proceedings-Vision, Image and Signal Processing*, *152*(5), 611–615. http://dx.doi.org/10.1049/ip-vis:20059022.

Wu, S., Zhong, S.-h., & Liu, Y. (2019). A novel convolutional neural network for image steganalysis with shared normalization. *IEEE Transactions on Multimedia*, *22*(1), 256–270. http://dx.doi.org/10.1109/TMM.2019.2920605.

Xia, P., Zhang, L., & Li, F. (2015). Learning similarity with cosine similarity ensemble. *Information Sciences*, *307*, 39–52.

Xu, G., Wu, H.-Z., & Shi, Y.-Q. (2016). Structural design of convolutional neural networks for steganalysis. *IEEE Signal Processing Letters*, *23*(5), 708–712. http://dx.doi.org/10.1109/LSP.2016.2548421.

Yang, J., Liu, K., Kang, X., Wong, E. K., & Shi, Y.-Q. (2018). Spatial image steganography based on generative adversarial network. http://dx.doi.org/10.48550/ARXIV.1804.07939, URL https://arxiv.org/abs/1804.07939.

Yang, C.-H., Weng, C.-Y., Tso, H.-K., & Wang, S.-J. (2011). A data hiding scheme using the varieties of pixel-value differencing in multimedia images. *Journal of Systems and Software*, *84*(4), 669–678. http://dx.doi.org/10.1016/j.jss.2010.11.889.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, *87*(9), 1423–1447. http://dx.doi.org/10.1109/5.784219.

Ye, J., Ni, J., & Yi, Y. (2017). Deep learning hierarchical representations for image steganalysis. *IEEE Transactions on Information Forensics and Security*, *12*(11), 2545–2557. http://dx.doi.org/10.1109/TIFS.2017.2710946.

Zhan, B., Zhou, L., Li, Z., Wu, X., Pu, Y., Zhou, J., et al. (2022). D2FE-GAN: Decoupled dual feature extraction based GAN for MRI image synthesis. *Knowledge-Based Systems*, *252*, Article 109362. http://dx.doi.org/10.1016/j.knosys.2022.109362, URL https://www.sciencedirect.com/science/article/pii/S0950705122006839.

Zhang, K. A., Cuesta-Infante, A., Xu, L., & Veeramachaneni, K. (2019). SteganoGAN: High capacity image steganography with GANs. http://dx.doi.org/10.48550/ARXIV.1901.03892, arXiv preprint arXiv:1901.03892.

Zhang, R., Dong, S., & Liu, J. (2019). Invisible steganography via generative adversarial networks. *Multimedia Tools and Applications*, *78*(7), 8559–8575. http://dx.doi.org/10.1007/s11042-018-6951-z.

Zhang, H., Ping, X., Xu, M., & Wang, R. (2014). Steganalysis by subtractive pixel adjacency matrix and dimensionality reduction. *Science China. Information Sciences*, *57*(4), 1–7. http://dx.doi.org/10.1007/s11432-013-4793-x.

Zhang, X., & Wang, S. (2006). Efficient steganographic embedding by exploiting modification direction. *IEEE Communications Letters*, *10*(11), 781–783. http://dx.doi.org/10.1109/LCOMM.2006.060863.

Zhang, R., Zhu, F., Liu, J., & Liu, G. (2019). Depth-wise separable convolutions and multi-level pooling for an efficient spatial CNN-based steganalysis. *IEEE Transactions on Information Forensics and Security*, *15*, 1138–1150. http://dx.doi.org/10.1109/TIFS.2019.2936913.

Zong, H., Liu, F.-l., & Luo, X.-y. (2012). Blind image steganalysis based on wavelet coefficient correlation. *Digital Investigation*, *9*(1), 58–68. http://dx.doi.org/10.1016/j.diin.2012.02.003.