Digital images steganography using adversarial embedding

Solène Bernard

CENTRALE LILLE

THÈSE

Présentée en vue d'obtenir le grade de

DOCTEUR

En

Spécialité: Automatique, Génie informatique, Traitement du Signal et des Images

Par

Solène Bernard

DOCTORAT DÉLIVRÉ PAR CENTRALE LILLE

Titre de la thèse

Stéganographie d'images numériques via l'utilisation de réseaux de neurones sous présence d'un adversaire

Digital images steganography using adversarial embedding

Soutenue le 25 octobre 2021 devant le jury d'examen :

Présidente,	Caroline Fontaine	DR CNRS, Laboratoire LMF, ENS Paris-Saclay
Rapporteur,	Andrew Ker	Professeur à Oxford University College
Rapporteur,	Laurent Amsaleg	DR CNRS, Laboratoire IRISA, Rennes
Examinateur,	Teddy Furon	CR INRIA, Laboratoire IRISA, Rennes
Examinatrice,	Luisa Verdoliva	Associate Professor, University Federico II of Naples, Italie
Directeur de thèse,	Patrick Bas	DR CNRS, CRIStAL, Université de Lille
Co-directeur de thèse,	John Klein	MdC, CRIStAL, Université de Lille
Co-directeur de thèse,	Tomas Pevny	Chercheur à CVUT, Prague

Thèse préparée dans le Laboratoire CRIStAL

Ecole Doctorale SPI 072

centralelille Université de Lille CRIStAL Centre de Recherche en Informatique, Signal et Automatique de Lille cnrs

Remerciements

L'exercice d'exprimer ma gratitude envers tous les gens qui m'ont tant apporté pendant cette aventure qu'est la thèse n'est pas une mince affaire, mes mots s'avérant ne jamais être à la hauteur de la reconnaissance que j'ai pour eux.

First of all thank you to the member of the jury Caroline Fontaine, Teddy Furon, Luisa Verdoliva and especillay to the reviewers Andrew Ker and Laurent Amsaleg for accepting to evaluate this manuscript. Thank you very much for all your advices, enlightenments and interesting interrogations which nourished this work.

Many thanks to Tomas for your outstanding supervision. I am glad that you shared with me your amazing ideas, sometimes around our weekly breakfast meeting in Avast. I am very lucky that you gave me the possibility to discover a bit of the spirit of Prague. It was an awaken dream to live on the Vltava river on a barge for a month, and to be able to have those snowy walks in the Letna park and short movie sessions in Cinema Svetozor. I am y glad for having met Leah, for all the parties and trips you organized so well (Neuschwanstein definitly worthed the trip!), Nklas, Antonio, Dominik, Michal, Honza, Tonda, Martin and Benjamin. You all made my trips memorable and special, and brought me a lot. Merci également à Ruben et Nathan, avec qui ont formait une petite bulle d'expats ressourçante.

Un immense merci à John et Patrick pour votre confiance et votre superbe supervision. J'ai eu la chance d'avoir des encadrants passionnés et aimant transmettre. Un redondant encore grand merci à Patrick pour son soutien indefectible. Merci également à Ingrid pour ses conseils couture et pour son chaleureux accueil lors des petites fêtes Sigma.

Evidemment, un grand merci à toute l'équipe Sigma avec qui j'ai adoré partager des réunions d'équipe, l'organisation du Gretsi, et bien sûr d'innombrables pauses cafés, IRL ou en ligne: Amine, Arnaud, Ayoub, Barbara, Christelle, Diala, Kevin, Jan, Jérémie, Michaël, Nouha, Ouafae, Patrick, Pierre, Pierre, Pierre-Antoine, Rémi, Rémy, Rui, Théo, Vincent, Wadih, Xiaoyi. Merci également à mes fantastiques co-bureaux Quentin, Markus, Yoann et Guillaume (et pour tous tes conseils !). Merci en particulier à Pierre, mon ancien professeur en troisième année à Centrale, pour ta bienveillance et tes encouragements.

Un immense merci à mes amis de lycée Alexis, Antoine, Elise, Eliot, Edouard, Fanny, Garance, Jeanne, Jean-Michel, Julie, Lucas, Noé, Paul, Pierre, Souleymen. J'ai adoré partager avec vous pendant ces trois ans un voyage frigorifique en Estonie, un voyage à Prague à se gaver de trodlos et de chocolat cachés, un festival fluo et oragesque, un voyage plongée en Martinique, un confinement à Concarneau, des vacances à Sainte-Maxime. Mais aussi nos moments plus banals non moins marquants, comme nos après-midi ensoleillés à Maisons-Alforts, les barbecues, les concours de cuisine, les séances films et séries. J'ai hâte de nos prochaines aventures, et de vous soutenir à mon tour dans vos projets passionnants.

Un énorme merci à ma coloc Emilie, qui m'a toujours supportée et soutenue, pendant des moments confinés pas forcément faciles. Également à Lauren qui a toujours sû me conseiller et trouver les mots pour me réconforter.

Merci à toute ma famille étendue qui m'a si bien soutenue lors de ma soutenance. Merci beaucoup à mon cousin Alexis qui m'a beaucoup rassurée dans les coups durs. Tes gentilles paroles restent encrées et me réconfortent au quotidien. Merci à mes parents qui nous ont toujours encouragés avec mon frère et ma soeur à suivre avant tout la voix qui nous passionne. Un grand merci à ma grande soeur Elsa pour toujours avoir une oreille disponible pour moi, et pour son soutien inconditionnel.

Finalement, un immense merci à Eliot avec qui j'ai hâte de partager de prochaines aventures.

Contents

Notations

Vectors

$\mathbf{x} = (x_i)_{1 \leq i \leq n}$	Cover image
$\mathbf{y} = (y_i)_{1 \leq i \leq n}$	Stego image
$\mathbf{b} = (b_i)_{1 \leq i \leq n}$	Modification vector
$\boldsymbol{\rho} = (\rho_i^j)_{1 \leq i \leq n, j \in \mathcal{B}}$	Cost matrix
$\boldsymbol{\pi} = (\pi_i^j)_{1 \leq i \leq n, j \in \mathcal{B}}$	Probability matrix

Sets

$\mathbb{N}$	Natural Numbers
$\mathbb{R}$	Real Numbers
$[\![0, n]\!]$	Set of natural numbers comprised between 0 and n
$\mathcal{B}$	Modification set
$\mathcal{I}$	Images set
$\mathcal{X}$	Cover set
$\mathcal{Y}$	Stego set
$\mathcal{M}$	Messages set
$\mathcal{K}$	Keys set

Other Symbols

N	Number of objects in the database
n	Number of pixels in an image

Functions

$[P]$	Iversion bracket, returning 1 iif P is true.
$\mathbb{1}_A(x)$	Indicator function, returning 1 iif $x \in A$.

Probability distributions

$\mathcal{P}_{\mathcal{X}}$	Probability distribution of covers
$\mathcal{P}_{\mathcal{Y}}$	Probability distribution of stegos
$\mathcal{P}_{\mathcal{M}}$	Probability distribution of messages
$\mathcal{P}_{\mathcal{K}}$	Probability distribution of keys

Introduction 0

The word steganography comes from ancient Greek with the combination of *steganós* ($\sigma\tau\epsilon\gamma\alpha\nu\sigma\sigma$), meaning "cover", and *grapho* ($\gamma\rho\alpha\phi\omega$) meaning "I write". Steganography is the practice of concealing a message discreetly within another content without arousing an analyst's suspicion. Steganography differs from cryptography in that cryptography conceals only the content of the message through encryption. Steganography conceals the presence of the message itself.

We can find ancient examples of back to 440 BC of use of steganography, but its utilization carries on today. It is used today in both legitimate and malicious usages. It allows civilians to communicate in an authoritarian state that controls the communications and allows the traffic of illegal content such as child pornography.

Electronic files containing multimedia content (such as text, audio signals, or images) are good candidates for cover for several reasons: (i) they are frequently and massively exchanged so their presence in a communication channel will not raise the alarm per se, (ii) they are large files where a secret can be more easily inserted, (iii) we can use computational intelligence and resources to design clever ways to perform the insertion. However, the technique of steganography entirely depends on the nature of the content. Hiding a message in a .mp3 is different from embedding it in a .pdf file. Therefore, the steganographer requires a deep understanding of the structure of those files. The subject of this thesis is to hide a message in digital images by slightly modifying an original image. It is what we call steganography by *cover* modification.

Chapter 1 offers a description of the usual representation of digital images and the JPEG format. Then we present a naive scheme for steganography to give intuition to the reader about how easy steganography can be and how it can be easily detectable.

Then we present the two steps on which rely the most common techniques used in steganography. The first one is to design a distortion function (which is often additive w.r.t. the image coefficients) to adapt the embedding of the message to the content of the cover, and thereby avoid risky areas (such as smooth textures like sky or walls), whose modification is highly detectable. The second part is to embed the message into the cover while minimizing the distortion function (which is the sum of the impact of embedding into each DCT coefficient individually). Those are two independent tasks that are research areas of their own. In this work, we only improve the first part, i.e. assigning a cost to each image coefficient. The second part is discarded, and we use a theorem providing us a way to simulate the optimal coding function (i.e. minimizing the additive distortion function) by

sampling a stego image.

Steganalysis, aiming at detecting the presence of a message in an image, has the antagonist role from steganography. It consists in extracting features which are relevant to discriminate the modified images from the original ones.

Practical steganographic security depends on the advances in steganalysis domain. To measure steganography security, we consider the worst-case attack by following Kerckhoffs's principle, stating that the steganalyst is aware of the steganographic technique, except the secret key. We finish the chapter by presenting the specific steganalysis tools relying on particular feature extractions, each adapted to detect a distinct embedding strategy.

Chapter 2 introduces the basics of machine learning. We first present its ambition of generalization by learning through examples. Convolutional Neural Networks (CNN) achieve outstanding performances in many computer vision tasks at the cost of designing a good architecture. The perform well in steganalysis too. They accurately discriminate cover from stego images and detect invisible-to-the-eye modifications applied to an image. This new method automatizes the traditional feature extraction step. It is replaced by the learning of parameters via an optimization algorithm.

Then we explain the importance of the differentiability of the operations within deep learning models because it is an essential ingredient for optimization algorithms.

We present at the end an overview of the existing architectures adapted to steganalysis. We implemented in this thesis some of those architectures to conduct experiments.

We conclude by a weakness of deep learning models. Malicious users who know the model can easily bypass a classifier. It is the ideal security breach that steganographic schemes can exploit.

Chapter 3, presenting our first contribution, starts with the following observation: for a steganographer aware of the steganalysis model, state-of-the-art offers adversarial embedding schemes to adapt the distortion function (attaching a cost to each image coefficient) to avoid the detection by a model. Nevertheless, the steganalyzer could reply by training a new classifier to detect the new technique. It leads to an endless game between two competitive players with opposite ambitions. The issue of the situation relies on the pair of actions of both rivals. In other words, each player wonders "*How can I anticipate what the other will play such as I behave optimally.*" This is the exact context of competitive games.

Our first contribution consists of introducing game theory notions to solve the steganographic game. The introduction of the problem from the game theory point of view leads to a formal definition of the best embedding function. Nash equilibrium [1] in a zero-sum game defines it as the one minimizing the optimal detector. Unfortunately, it is a min max optimization problem over infinite sets of actions, where the objective function has no explicit expression. We propose a practical

[1] We do not formally play the Nash equilibrium because we only considered in this work pure strategies.

solution by constructing an iterative game, where at each step, players play a new action. The set of actions is consequently increasing, but finite, such as the game is solvable at each step. At each iteration comes the creation of (i) a new distortion function defeating the optimal over a finite number of detectors and (ii) a new detector to identify the new stegos. By doing so, we show that the game converges and that steganography is improved. To do the experiments, we used ADV-EMB, which proposes a heuristic to avoid a detector.

Chapter 4 consists in improving the resolution of the min max problem that ADV-EMB solves in our first contribution with a heuristic. We propose to remove the heuristics and to use a more powerful optimization technique i.e. gradient descent. Because we are simulating the embedding, it requires us to compute the gradient of the expectation of the detectability of the optimal detector with respect to probability distribution parameters. We show how to approximate such gradient by doing continuous (instead of discrete) modifications to the cover thanks to the Gumbel distribution while complying with an entropy constraint. We provide an iterative gradient descent more powerful than ADV-EMB as it can defeat a set of classifiers.

Contributions

Journal

2021 *Backpack, a differentiable adversarial embedding scheme*, to be submitted to IEEE TIFS

2020 Solène Bernard, Patrick Bas, John Klein, Tomáš Pevný, *Explicit optimization of a* min max *steganographic game* [1]. IEEE Transactions on Information Forensics and Security, Institute of Electrical and Electronics Engineers.

Conferences

2019 Solène Bernard, Tomáš Pevný, Patrick Bas, John Klein, *Exploiting adversarial embedding for better steganography* [2]. IH-MMSec, Jul 2019, Paris, France.
Best student paper award

2019 Solène Bernard, Tomáš Pevný, Patrick Bas, John Klein, *Utilisation d'insertions adverses pour améliorer la stéganographie* [3]. GRETSI, Aug 2019, Lille, France.

2021 Solène Bernard, Patrick Bas, John Klein, Tomáš Pevný, *Optimizing Additive Approximations of Non-Additive Distortion Functions* [4]. 9th ACM Workshop on Information Hiding and Multimedia Security, Jun 2021, Brussels (online), Belgium.
Best student paper award

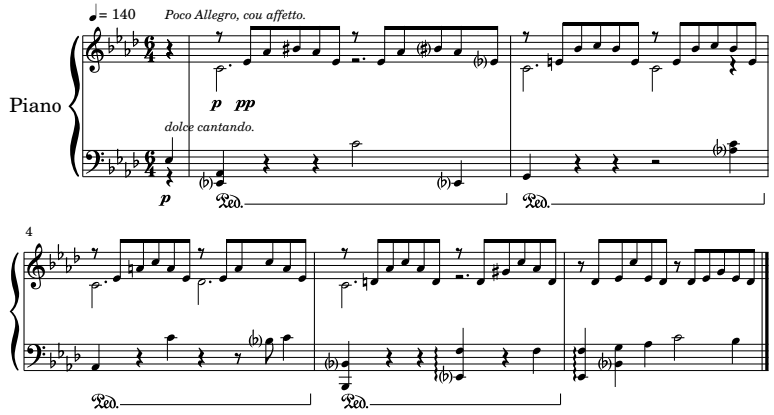
Reproductible research

2021 `https://gitlab.univ-lille.fr/solene.bernard/backpack`

1 Basics of steganography and steganalysis

In November 1940 in France, the Vichy Government installed the *Service des Contrôles Techniques* in charge of watching French people through their letter correspondences. Eve, an employee of the SCT, opens an envelope and finds a letter with a music sheet attached.

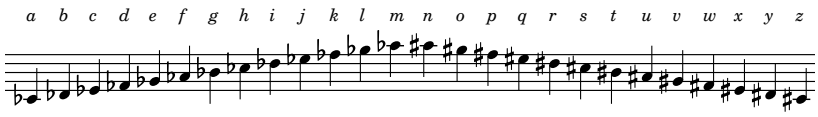
> My Dearest Bob,
> I would be glad if you could share with me your thoughts about the music I wrote.
> Yours always,
> Alice



Click here to listen to the music.

She quickly glances at it, closes it and classifies it as *non-sensitive letter*, and stacks it on the corresponding pile in order to send it to Bob as Alice expected.

Once Bob receives it, he follows the rule that he agreed with Alice. He reads all notes preceded by a sharp or flat symbol, then converts it to a letter of the Latin alphabet according to the following decoding table:



Stacking all those letters in the order of appearance in the sheet (from left to right, both of staves at the same time) leads to the decoded message "*attack leave*". As soon as Bob discovers it, he understands that his security is at stake and immediately leaves for another shelter.

In this example, Alice modifies the five first measures of the piece Liebesträume composed by Liszt to embed her message.

This fictional example illustrates the concept of *steganography* and *steganalysis*. Since the formal definition of the prisoners' problem by Simmons [5] in 1983, steganography and steganalysis have been considered as a *hide and seek* game.

The prisoners' problem tells the story of two accomplices in a crime locked in separated cells. They want to coordinate an escape plan, but a warden watches their communication. The guardian will only permit the exchanges if the information contained in the messages is innocuous. The prisoners have to deceive the warden by finding a way of communicating secretly in the exchanges, i.e., of establishing a *subliminal channel* between them.

Steganography is the art of dissimulation. When *Alice* wants to communicate a secret message to *Bob*, steganography consists of hiding a message in a *content*, while Alice and Bob previously shared the decoding method. There are limitless possibilities for the nature of the content as long as it looks legitimate: it could be a music sheet or a poem, an image.

Steganalysis is the adversarial act of detecting if a message is hidden in a content. *Eve*, as in the expression *eavesdropping*, wants to stop Alice and Bob from keeping secrets from her. She has the power to cut the communication channel between the two of them, which encourages them to be the least suspicious as possible.

History is full of amusing anecdotes about past usages of steganography. To our knowledge, the first utilization goes back to ancient Greece. In the book *Histories* written by Herodotus, Histiaeus tattoos a secret message on the skull of a slave. The messenger goes undercover when his hair grows back, and Aristogoras decodes the message by simply shaving him.

Another famous use of steganography is writing using invisible ink on regular paper. The ink can be made of lemon juice, and the message can be read by exposing the paper to a heat source.

In this thesis, we will limit ourselves to a particular type of content: digital images. The following section aims at introducing the image format before explaining how to apply steganography in images.

1.1 Image representation

1.1.1 *From photons to digital data*

Photography, which means etymologically "coloring with light" is the ambitious challenge to convert into an object the environment which created a physical sensation in a human.

This technique is more than standard today, as it is part of our daily lives as all of our smartphones can take pictures. However, until the last century, photography was not ordinary, requiring considerable technological and scientific progress.

Two dominant photographic sensors exist: CCD (Charge-Coupled Device) and CMOS (Complementary Metal Oxide Semiconductor) using the photoelectric effect. It quantifies the number of photons hitting a photographic cell array to translate it to numerical data.

In order to reproduce coloured photography, researchers first looked for the biological composition of the human eye. It is in 1802 that Young discovered [8] that it exists three types of photoreceptors (now

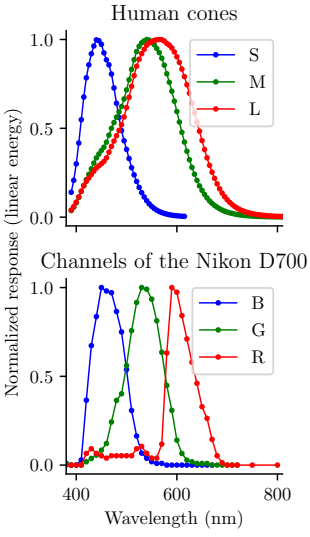


Figure 1.1: Normalized response of the three cones S, M, L of the human eye [6] (top) and of the three color channels of a Nikon D700 device [7] (bottom).

known as cone cells) in the human eye, each of which are sensitive to a particular range of visible light, named short, medium, or large given their size.

A few years later, Maxwell demonstrated theoretically in 1855 [9] that any monochromatic light stimulating three receptors should be able to be equally stimulated by a set of three different monochromatic lights. This would mean that a superposition of three colours could reproduce every sensation of colour, therefore called *primary colours*. Therefore, the first colour photography was produced by taking pictures three times of the same scene with three coloured filters.

Today, the representation of colour digital images still relies on the superposition of three colour channels: red, green, and blue, so three types of sensors are used. Figure 1.1 shows both absorption spectrums of receptors in the eye and in a Nikon D700 camera look alike.

In order to take at one instant a picture with three kinds of photoreceptors, the most common solution is to use a colour filter array (CFA). It is a mosaic of tiny colour filters placed over the pixel sensors of an image sensor to capture colour information. Multiple subjective designs of the CFA exist. The most popular one is the Bayer Filter, plotted in Figure 1.2.

The raw image data captured by the image sensor is then converted to a full-colour image (with intensities of all three primary colours represented at each pixel) by a demosaicing algorithm which is tailored for each type of colour filter.

Gray scale images are coded only with one channel. It contains the luminance Y, which is equal to a linear combination of the three color channels R, G and B:

$$Y = 0.299R + 0.587G + 0.114B \tag{1.1.1}$$

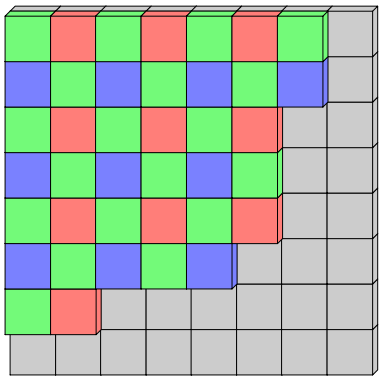


Figure 1.2: The Bayer CFA. Each two-by-two submosaic contains 2 green, 1 blue, and 1 red filter, each filter covering one pixel sensor.

Figure 1.3: Three colors (left) channels, which when superposed produce a color image (fourth image). The last image is the luminance computed with Equation 1.1.1.

In this thesis, we will work only with the luminance channel, i.e., with grayscale images. Images are therefore coded with an integer in a specific range of values. For example, 8-bits coded images have pixels coded with an integer between 0 and 255. The natural representation is the spatial representation that is given at the output of the camera sensor. However, the research area of image classification or pattern

recognition motivated the challenge of finding a new representation of images where the content of images can be represented in fewer features than the number of pixels in the image. This goal is close to the one of data compression, which favours a sparse representation. It is essential in this thesis to present the most famous compression algorithm JPEG because we will work with that kind of image. Nevertheless, we will firstly present the DCT transform.

There were many suggestions of transforms suggested by Karhunen-Loève [10], Fourier [11] or Welsh/Hadamard [12]. The most popular is the Discrete Cosine Transform [13], proposed by Nasir Ahmed in 1972 and can be computed by using a fast algorithm.

1.1.2 *Discrete Cosine Transform (DCT)*

For an 8×8 block of luminance values $\mathbf{x}[i, j]$, $i, j = 0, \dots, 7$, the 8×8 block of DCT coefficients $\mathbf{d}[k, l]$, $k, l \in [\![0, 7]\!]$ is computed as a linear combination of luminance values

$$\mathbf{d}[k, l] = \sum_{i,j=0}^{7} \mathbf{f}[i, k]\mathbf{f}[j, l]\mathbf{x}[i, j], \tag{1.1.2}$$

where $\mathbf{f}[i, k] = \frac{\mathbf{w}[k]}{2} \cos \frac{\pi}{16} k(2i + 1)$, $\mathbf{w}[0] = 1/\sqrt{2}$ and $\mathbf{w}[k > 0] = 1$. The coefficient $\mathbf{d}[0, 0]$ is called the DC coefficient while the remaining coefficients are called AC coefficients. The DCT is inversible, and its inverse called IDCT is given by

$$\mathbf{x}[k, l] = \sum_{i,j=0}^{7} \mathbf{f}[k, i]\mathbf{f}[l, j]\mathbf{d}[i, j]. \tag{1.1.3}$$

The DCT can be interpreted as a change of basis, where the unit vectors are 64 units images of size 8×8.

Those 64 unit images indexed by i, j are given by $\mathbf{F}_{i,j}[k, l] = \mathbf{f}[k, i]\mathbf{f}[l, j]$, they are plotted on Figure 1.4. An example of decomposition of a spatial block is shown on Figure 1.5.

It is shown on section 2.2.2 that this operation can be interpreted as 64 operations of convolution with a kernel of size 8×8, with a stride of 8.

DCT transform is the core of the JPEG compression, whose purpose is to reduce the number of bits needed to code the image at a price of data loss.

1.1.3 *JPEG compression*

The first step of JPEG compression is to apply DCT transform to each 8×8 (non-overlapping) block of the spatial image.

Then there is a quantization step, which is the step which produces data loss. During quantization, the DCT coefficient $[i, j]$ is divided by quantization step $\mathbf{q}[i, j]$ from the quantization matrix $\mathbf{q}$ and rounded[1] to their closest integers:

$$\mathbf{D}[i, j] = \text{round}\left(\frac{\mathbf{d}[i, j]}{\mathbf{q}[i, j]}\right), \forall i, j \in [\![0, 7]\!]. \tag{1.1.4}$$

[1] Depending on the implementation, the rounding operation in Equation 1.1.4 can be changed.

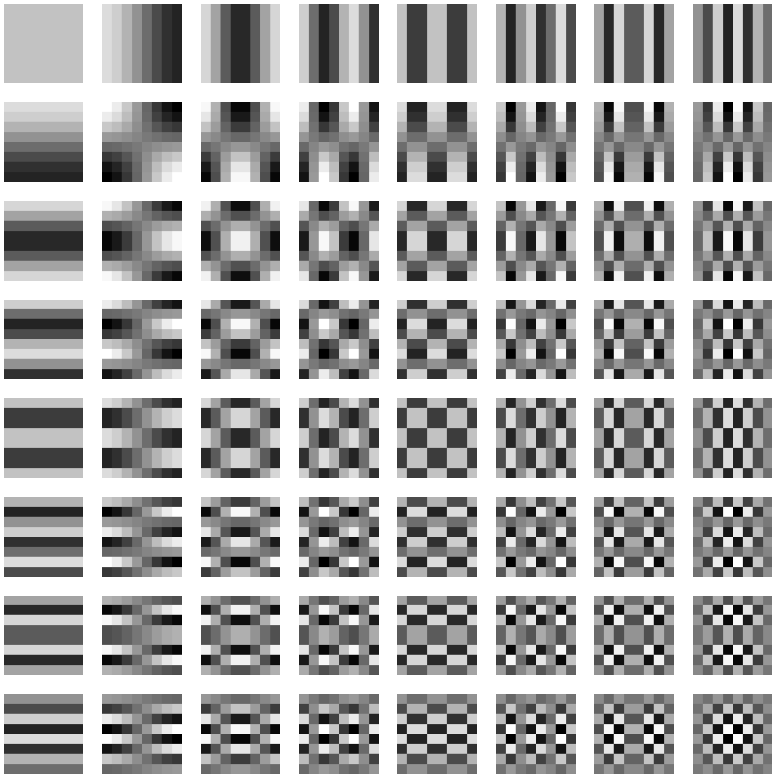


Figure 1.4: Visualization of the 64 DCT Filters $\mathbf{F}_{i,j}$, where i is for the row and j for the column. Horizontal frequencies increase from left to right (with increasing j), and vertical frequencies increase from top to bottom (with increasing i). The constant-valued basis function at the upper left is often called the DC basis function, and the corresponding DCT coefficient $\mathbf{d}[0,0]$, the DC coefficient.

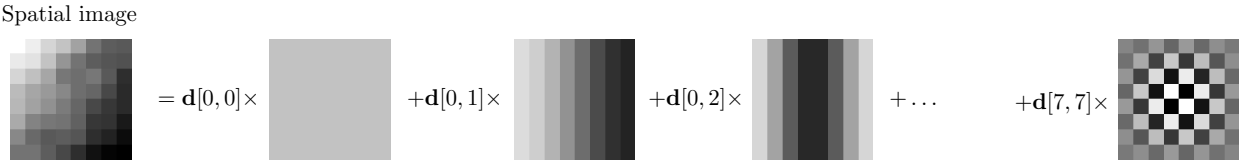


Figure 1.5: Decomposition of a 8×8 spatial image into a linear combination of 64 DCT filters.

The larger are the quantization steps, the fewer bits are needed to code all the image coefficients, but the more the image quality is damaged.

The JPEG standard recommends a set of quantization matrices indexed by a quality factor f which is an integer between 1 and 100. They are given for the luminance channel by:

$$\mathbf{q}_f[i,j] = \text{floor}\left(\frac{50 + S \times \mathbf{q}_{50}[i,j]}{100}\right) \text{ with } S = \begin{cases} 200 - 2f & \text{if } f > 50 \\ 5000/f & \text{if } f \leq 50. \end{cases} \tag{1.1.5}$$

In order to decompress a JPEG image, one must do the inverse steps of the compression in the reverse order, except the rounding step because it is an irreversible operation. The decompression operation, noted JPEG^{-1}, is composed of the dequantization of the image (i.e. multiplying each coefficient by the corresponding quantization step) then the application of the IDCT. The more the quality factor is close to 100, the more the decompressed image is close to the original image, as shown in Figure 1.6. We can see in Figure 1.7 an original image of size 32×32, then the decompressed versions of this image compressed

Example of quantization matrix for luminance at quality factor 75 and 95:

$$\mathbf{q}_{75} = \begin{pmatrix} 8 & 6 & 5 & 8 & 12 & 20 & 26 & 31 \\ 6 & 6 & 7 & 10 & 13 & 29 & 30 & 28 \\ 7 & 7 & 8 & 12 & 20 & 29 & 35 & 28 \\ 7 & 9 & 11 & 15 & 26 & 44 & 40 & 31 \\ 9 & 11 & 19 & 28 & 34 & 55 & 52 & 39 \\ 12 & 18 & 28 & 32 & 41 & 52 & 57 & 46 \\ 25 & 32 & 39 & 44 & 52 & 61 & 60 & 51 \\ 36 & 46 & 48 & 49 & 56 & 50 & 52 & 50 \end{pmatrix}$$

$$\mathbf{q}_{95} = \begin{pmatrix} 2 & 1 & 1 & 2 & 2 & 4 & 5 & 6 \\ 1 & 1 & 1 & 2 & 3 & 6 & 6 & 6 \\ 1 & 1 & 2 & 2 & 4 & 6 & 7 & 6 \\ 1 & 2 & 2 & 3 & 5 & 9 & 8 & 6 \\ 2 & 2 & 4 & 6 & 7 & 11 & 10 & 8 \\ 2 & 4 & 6 & 6 & 8 & 10 & 11 & 9 \\ 5 & 6 & 8 & 9 & 10 & 12 & 12 & 10 \\ 7 & 9 & 10 & 10 & 11 & 10 & 10 & 10 \end{pmatrix}$$

at different quality factors. For low-quality factors, artefacts due to 8×8 block-wise DCT transform are apparent.

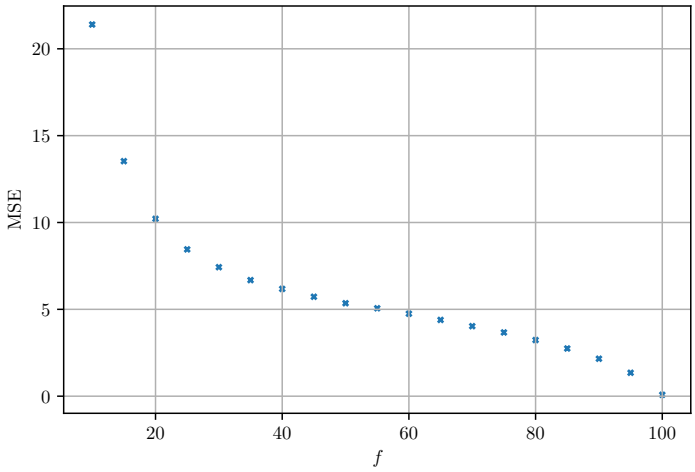


Figure 1.6: Mean Square Error (MSE) of the compression error, depending on the quality factor f. $\mathrm{MSE}(\mathbf{x}) = \sum_{i,j} x[i,j]^2$, and the compression error is equal to $\mathbf{x} - \mathrm{JPEG}^{-1}(\mathrm{JPEG}(\mathbf{x}))$. Note that the MSE for $f = 100$ is not equal to 0.

Figure 1.7: Vizualization of effect of image JPEG compression for three quality factors, f = 25, 50 and 100.

The two only representations used in this thesis are *spatial* and *JPEG*. Embedding a message in an image depends on its representation. We will see in the next section the very basics of steganography by introducing a simple example in the spatial domain.

1.2 Naive steganographic schemes

1.2.1 *LSB replacement*

A very straightforward steganographic scheme is LSB replacement. For a grayscale digital image where a binary number codes the value of each pixel (for example with 8-bits code integer between 0 and 255), the Least Significant Bit is the last bit of the pixel (as the representation is binary, the LSB contains the parity of the pixel). LSB replacement consists of hiding the message bits in the LSBs of pixels, such as at

maximum the pixel is modified by $+1$ when it is even, and by -1 when it is odd. Figure 1.8 shows the illustration of an original image containing 8 pixels $(70, 73, 77, 81, 79, 75, 75, 79)$ coded on 8 bits, and for which the LSBs are replaced by the 8 message bits $(1, 1, 0, 0, 1, 1, 1, 0)$.

Original image (Base 10)	Original image (Base 2)	Message (Base 2)	Modified image (Base 2)	Modified image (Base 10)
70	0100011**0**	**1**	0100011**1**	**71**
73	0100100**1**	**1**	0100100**1**	73
77	0100110**1**	**0**	0100110**0**	**76**
81	0101000**1**	**0**	0101000**0**	**80**
79	0100111**1**	**1**	0100111**1**	79
75	0100101**1**	**1**	0100101**1**	75
75	0100101**1**	**1**	0100101**1**	75
79	0100111**1**	**0**	0100111**0**	**78**

Figure 1.8: Example showing the embedding of a binary message of length 8 in a same-sized cover, by replacing the Least Significant Bit of each pixel in the cover (red bits) by the message bit (blue bits). Finally, 4 pixels have been modified (bold pixel values in the stego array).

Because we only allow modifications on the LSB of the pixels, we assume that the human eye will not detect the change. The assumption is still widely used in all steganography schemes: with a wide range of shades (for example here, 8 bits coded images for which the range is a size of $2^8 = 256$, from black to white), adding -1 or $+1$ to some pixels in a spatial image, whatever the quantity, is assumed to have no impact on the detectability of a human eye.

In Figure 1.9, we can easily observe what would happen if we use more significant bits to hide the message. Alice can replace the two, three, ..., seven, or eight least significant bits to embed more information in the same content). For the 8-LSB, this means replacing all the image bits with the message bits. Even the 2-LSB replacement method creates embedding artefacts in the sky that are highly suspicious.

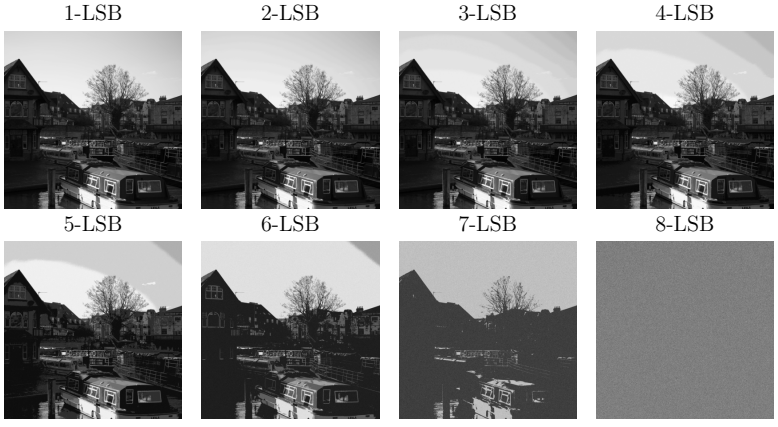


Figure 1.9: Effect on the stego image of embedding with i-LSB replacement with different depths, from $i = 1$ to 8.

This example highlights the main compromise of steganography: *hiding as much data as possible, while being the least detectable.* Those are two adversarial challenges, as we suppose that hiding more data will lead to higher detectability and vice versa.

Given Figure 1.9, replacing only the least significant bit seems to

be the maximum Alice can do to remain undetectable. However, it is not enough, as we can design an elementary steganalysis detector able to detect this kind of embedding, as will be shown in the next section 1.2.2. We insist that the steganalyzer is very simple to point out how naive this steganographic scheme is. Then, in section 1.2.3, we will show the importance of the embedding scheme by designing another one that reduces the number of modifications applied to the cover.

1.2.2 *Importance of respecting natural statistics*

Even in the LSBs, there is information on the image content which must be preserved. *Natural* images have a particular distribution for two reasons: first (*i*) because of some common characteristics of the object in the picture, and second (*ii*) because the camera introduces a particular noise that can be modelled. Let us focus only on the first reason. For example, Figure 1.10 shows a cover image (top left image), and then, on the right, is plotted its LSBs (which are binary numbers, hence only two colours are used, black and white). The sky area includes long sequences of pixels with the same parity. Hiding a message in those areas is risky, as it introduces some noise in a usually smooth area, as one can see in the red square zoomed areas. With the same logic, hiding a message in the tree area (green squares) seems less dangerous, as the hidden message could be interpreted as the natural noise of the image. The different distribution of LSB in smooth areas gives us the intuition that a steganalyzer can be easily designed.

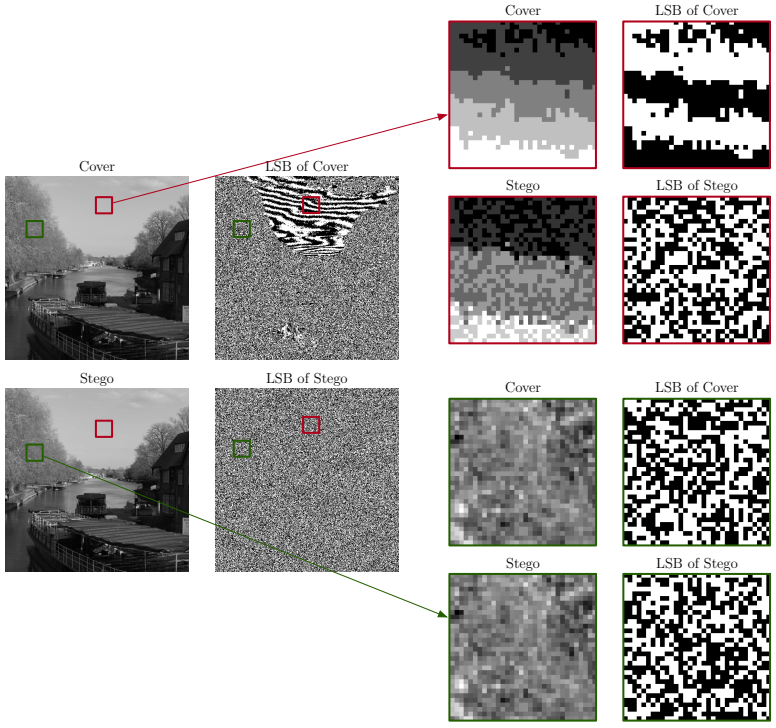


Figure 1.10: Visualization of the impact of the insertion of a message of length n^2 inside an image of size $n \times n$. LSBs are plotted by representing a black or white pixel for an even or odd value of the pixel (i.e. LSB equals to 0 or 1). The eight images on the right are zooms of two regions of size 32×32 plotted on the left by coloured squares: a relatively smooth region in the sky (red) or textured region in a tree (green). We tuned the dynamic of colours in the zoomed images to improve visualization.

We can imagine a steganalysis detector inspired by the Histogram

attack proposed in [14], which compares, in the LSB plane, the ratio of equal adjacent pixels compared to the ratio of different adjacent pixels. So in other words, it computes the following quantities:

- the ratio of two adjacent with the same parity, and
- the ratio of two adjacent pixels with different parity.

Indeed, most natural images contain smooth areas, like the sky or house walls, where many consecutive pixels are equal, so the first ratio is expected to be higher than the second one. In contrast, for the modified images, there are no such areas. The chance that two consecutive pixels have the same parity depends directly on the probability distribution of the message, as the LSBs are equal to the message. For a message where each bit has a chance 1/2 of being equal to 0 or 1, the two ratios are equal to 1/2. In Figure 1.11, we insert a random message in several images (stegos images) by LSB replacement and plot the ratio of adjacent pixels[2] with different parity, compared to the ratio in the original images (cover images). We observe that in most natural images, the ratio is below 0.5, but for all stego images, it is equal to 0.5. It is a satisfying primary steganalyzer tool that only consists of finding a suitable threshold.

[2] For the experiment, we look at horizontally adjacent pixels.

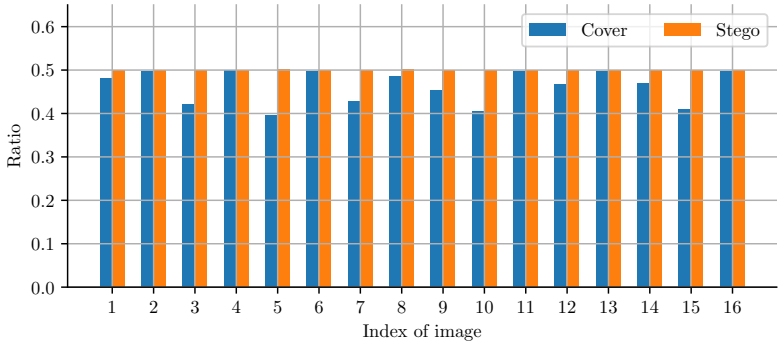


Figure 1.11: Ratio of adjacent pixels with different parity in 16 original image (blue bars) and in their 16 stegos version (orange bars). Two pixels, represented as squares, are adjacent when they share a common vertical vertice.

LSB replacement needs to be improved to preserve some elementary statistics (as the occurrence of continuities between adjacent pixels). The quality of preservation is measured through a so-called *distortion function* which is minimal when the model is preserved. The definition of distortion functions is itself a challenge. The following one is embedding a message while reaching the minimum of the distortion. It depends on the *coding method* (or *coding function*).

1.2.3 *Importance of the coding function*

The coding method is the pair of embedding and extracting methods that Alice and Bob previously agreed on to hide and read the message inside the image. If Alice wants to embed a message of m bits in an image of n pixels, the coding method of LSB replacement consists of replacing the first m LSBs of the image with the message bits and leaving the last $n - m$ bits unchanged. Note that the secret key often

contains an integer which is the seed of pseudo-random permutation aiming at shuffling the cover image so that m bits randomly located in the original image are changed[3]. On average, a pixel has a probability of $\frac{m}{2n}$ of being changed (which is called rate of modification). Section 1.4.1 present the general definitions of those objects.

Alice wants to minimize the embedding impact while embedding. For simplicity, it is measured by the number of modifications, as she thinks that the least number of changes, the best it is. It is an empirically defined distortion function. LSB replacement is not the most efficient coding method w.r.t. this distortion among the coding methods embed in the LSBs of pixels. The following pair of embedding and extracting functions achieve embedding of the same amount of message bits while doing fewer changes on average. It stands on the following idea: instead of hiding the message in the first pixels, one can embed the message *as a whole*.

First, let's take a very simple example where the message $\mathbf{m} = (m_0, m_1)$ is of size 2 and the image $\mathbf{x} = (x_0, x_1, x_2)$ contains 3 pixels. We here consider only the LSB of pixels, so $\mathbf{x} \in \{0,1\}^3$. The image containing the message is $\mathbf{y} = (y_0, y_1, y_2)$.

We recall that LSB replacement consists of replacing the LSB of the two first pixels x_0 and x_1, such as

$$\begin{cases} y_0 = m_0 \\ y_1 = m_1 \end{cases}$$

is the coding and decoding rule, so the image containing the message is equal to $\mathbf{y} = (m_0, m_1, x_2)$ (the last pixel is never changed).

Now let's see a better scheme where the decoding rule is:

$$\begin{cases} m_0 = y_0 + y_1 \\ m_1 = y_1 + y_2 \end{cases}, \tag{1.2.1}$$

where the + operation is for binary numbers, such as it is equivalent to the *xor* operation. This has the advantage to make Alice do at most 1 change. Indeed, there are four equiprobable possibilities for the value of the pair $(x_0 + x_1 - m_0, x_1 + x_2 - m_1)$:

$$\begin{pmatrix} x_0 + x_1 - m_0 \\ x_1 + x_2 - m_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \text{ or } \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ or } \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ or } \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

When it is equal to null vector, no change needs to be made as $\mathbf{x}$ already carries the message and $\mathbf{y} = \mathbf{x}$ in this case. For the three other cases, only one pixel needs to be changed, and the encoding rule can then be written as:

[3] In practice, Alice and Bob share a secret integer k. It permits both to generate the same pseudo-random permutation of the set of pixels. For example, NumPy function `np.random.seed(k)` can be used before calling a random permutation. Before Alice embeds the image, she shuffles the image coefficients according to the permutation generated with k. When Bob receives the image, it can apply the same permutation, such as he decodes the correct message.

$$\mathbf{y} = \begin{cases} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} & \text{if} \quad \begin{pmatrix} x_0 + x_1 - m_0 \\ x_1 + x_2 - m_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ \begin{pmatrix} x_0 \\ x_1 \\ x_2 + 1 \end{pmatrix} & \text{if} \quad \begin{pmatrix} x_0 + x_1 - m_0 \\ x_1 + x_2 - m_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ \begin{pmatrix} x_0 + 1 \\ x_1 \\ x_2 \end{pmatrix} & \text{if} \quad \begin{pmatrix} x_0 + x_1 - m_0 \\ x_1 + x_2 - m_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ \begin{pmatrix} x_0 \\ x_1 + 1 \\ x_2 \end{pmatrix} & \text{if} \quad \begin{pmatrix} x_0 + x_1 - m_0 \\ x_1 + x_2 - m_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \end{cases}.$$

This encoding technique is called *matrix embedding using binary Hamming codes* and can be generalizable for every message hidden in a image of $n = 2^m - 1$ pixels. Without detailing the exact method, Hamming matrices $\mathbf{H}$ are of the size $m \times (2^m - 1)$, and were first designed by Hamming in 1950 via Hamming codes [15], which are a family of linear error-correcting codes. Their construction has the good property that it always allows Alice to embed the message by changing the value of exactly one pixel with probability $1 - 1/2^m$, and not to change any pixel with probability $1/2^m$. This leads to a modification rate of $(1 - 1/2^m)/n = 1/(n + 1)$, which is substantially better than the previous rate of $m/2n$, as summarized in Table 1.1.

The Hamming matrix-matrix used in the example of Equation 1.2.1 is $\mathbf{H} = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$

For LSB replacement, each coefficient has a chance of m/n to embed a bit, and if it does, it has a chance $1/2$ to be modified. This gives an average modification rate of $m/2n$.

LSB replacement	Hamming code
$\frac{m}{2n}$	$\frac{1-1/2^m}{n}$

Table 1.1: Average modification rate (i.e. number of modification per pixel) obtained by either LSB replacement or matrix embedding using hamming codes, with $n = 2^m - 1$.

Finally, we designed an encoding function that gives a lower distortion than LSB replacement for a fixed couple (m, n) if $n = 2^m - 1$, where the distortion is defined as the number of modifications. Hence, the natural questions follow: could better encoding/extracting functions be designed? Does it depend on the chosen distortion function? In other words, is there a theoretical boundary of the performance of the encoding/extraction functions, w.r.t. a chosen distortion function? We first need to introduce some general definitions in section 1.3, and the answers will be exposed in section 1.4.4.

1.3 Information theoretic measures for steganography and steganalysis

At the beginning of the 20th century, telegrams were the primary medium of long-distance communication, but it was costly. In order to reduce the costs, data compression was an exciting challenge because its purpose is to transmit the same information but with a shorter sequence of characters. We will show in the next section that Shannon introduced a significant result in data compression, which is firmly tight to our problem, as embedding information in images can

be seen as a kind of compression.

1.3.1 *Information theory and Shannon's Theorem*

Shannon was trying to solve the question "how much information is contained in some piece of data?". He proposed a solution by finding how many bits of information can code a binary sequence of length n. To do so, he first introduced the notion of entropy:

Definition 1.3.1 (Entropy of a first-order source X)**.** *Consider an alphabet $A = \{a_1, \cdots, a_q\}$, and $(X_1, \cdots, X_n)$ is a tuple of n iid realizations of the first-order source X, such as $\forall i \in \{1, \cdots, q\}$, we denote by π_i the probability of X takes the value a_i, so $P(X = a_i) = \pi_i$. We define the entropy of the source X as*

$$H(\pi) = -\sum_{i=1}^{q} \pi_i \log_2(\pi_i). \tag{1.3.1}$$

This notion is often qualified as measuring the "amount of uncertainty" in a probability distribution. Nearly deterministic distributions (where the outcome is nearly sure) have low entropy; distributions closer to uniform have high entropy.

Shannon built this measure to write his central theorem [16], which stands that the length of any binary coding function of a set of n realizations of X is at least $nH(\pi)$ in average. In other words, it gives an average of the number of bits needed to encode symbols drawn from a distribution π. However, equivalently, this gives an upper bound of the data transmitted by a single realization of a distribution π, equal to $H(\pi)$.

Entropy is a significant notion in steganography as the entropy of the probability distribution over the stego images determines the number of message bits one can embed in it. Hiding a message of m bits in a cover content requires finding a distribution π over the set of stego images $\mathcal{Y}$ such as $H(\pi) = m$.

Nevertheless, steganography is not only about embedding a message; it is also to be undetectable. The following introduces the theoretical measure of steganography security.

It can also be shown that $H(\pi) \leq \log_2 q$ by concavity of the logarithm, and this upper bound is achieved when $\pi_1 = \cdots = \pi_q = 1/q$. See for example Figure 1.12 for $q = 2$.

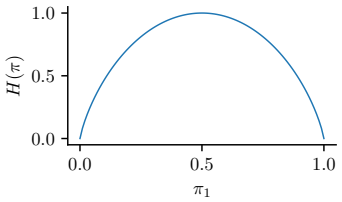


Figure 1.12: Entropy $H(\pi)$ for an alphabet $A = \{a_1, a_2\}$ of length 2, w.r.t. $\pi_1 = P(X = a_1)$

1.3.2 *Divergence of Kullback-Leibler*

If we have two probability distributions $P(x)$ and $Q(x)$ over the same random variable X, we can measure how different these two distributions are using the Kullback-Leibler (KL) divergence:

$$D_{\mathrm{KL}}(P||Q) = \mathbb{E}_{x\sim P}\left[\log \frac{P(x)}{Q(x)}\right] = \mathbb{E}_{x\sim P}[\log P(x) - \log Q(x)]. \tag{1.3.2}$$

The KL divergence has the property of being equal to 0 if and only if P and Q are identical distributions.

Perfect steganography should preserve the distribution of the cover images. In this context, Christian Cachin in 1998 [17] proposed a

distance based on Kullback-Leibler between the stego and the cover distributions to determine the security of the steganographic method. If the KL divergence is equal to 0, the warden cannot differentiate the stego images from the cover ones since both distributions for stego and cover objects are identical. However, it is not feasible in practice to design such a steganographic system under this strict bound because it is not a trivial task to determine the distribution of the cover images.

The following section presents the practical elements of steganography and steganalysis.

1.4 The stego game in practice

1.4.1 *Steganographic scheme*

In Fridrich's work of modern steganography [14], the possible steganographic methods are divided into three categories: *cover selection, cover synthesis* and *cover modification*. When a steganographer possesses a large natural image database, he can select the most suitable cover for data embedding, for example, by choosing the one that needs the least number of modifications to embed the message. The steganographer could also artificially synthesize a secret message into a texture image. In this report, we will only focus on steganography by *cover modification*. In this scenario, Alice chooses an initial image, which is called *cover* and modifies it in order to embed a secret message to obtain a so-called *stego* image.

Both Alice and Bob share a secret key $\mathbf{k}$ and a pair of function Emb and Ext which are necessary to *embed* in the cover and *extract* from the stego. So it is natural to define the following objects:

$$\begin{cases} \mathcal{I} \text{ set of image objects} \\ \mathcal{K}(\mathbf{x}) \text{ set of stego keys for } \mathbf{x} \in \mathcal{I} \\ \mathcal{M}(\mathbf{x}) \text{ set of all message that can be embeded in } \mathbf{x} \in \mathcal{I} \end{cases}, \quad (1.4.1)$$

and

$$\begin{cases} \text{Emb}: & \mathcal{I} \times \mathcal{K} \times \mathcal{M} \to \mathcal{I} \\ \text{Ext}: & \mathcal{I} \times \mathcal{K} \to \mathcal{M} \end{cases}, \quad (1.4.2)$$

such as $\forall \mathbf{x} \in \mathcal{I}, \forall \mathbf{k} \in \mathcal{K}(\mathbf{x}), \forall \mathbf{m} \in \mathcal{M}(\mathbf{x})$:

$$\text{Ext}(\text{Emb}(\mathbf{x}, \mathbf{k}, \mathbf{m}), \mathbf{k}) = \mathbf{m}. \quad (1.4.3)$$

We assume that the cover and stego objects have already been subject to some key-dependent permutation for simplicity of notation. We omit the dependence on the key from now on.

We denote $\mathcal{X} \subset \mathcal{I}$ the set of cover images and $\mathcal{Y}(\mathbf{x}) = \{\text{Emb}(\mathbf{x}, \mathbf{m}), \mathbf{m} \in \mathcal{M}(\mathbf{x})\}$ the set of stegos images that can be obtained from $\mathbf{x} \in \mathcal{X}$.

Most steganographic schemes work with a representation of cover and stego using finite set of symbols from some alphabet Σ using a symbol assignment function $\text{Symb} : \mathcal{X} \to \Sigma$. Here, symbols are a q-ary

alphabel $\Sigma = \{0, 1, 2, \cdots, q-1\}$ and all operations will be performed in $\mod q$ arithmetic, so that Symb is the function equals to $x \mapsto x \mod q$. Let cover and stego objects be represented as vectors $\mathbf{x} \in \Sigma^n$, $\mathbf{y} \in \Sigma^n$ respectively, with n being the number of pixels. Therefore, images are represented as one dimensionnal objects, and image elements can be indexed by an integer i comprised between 1 and n. The communicated message is also a q-ary vector $m \in \Sigma^m$ with $m < n$ being the length of the message.

Another important point is to define which protagonist knows what about the situation. Kerckhoffs introduced in *La cryptographie militaire* [18] in 1883 the basics of security of cryptology which is that everybody is supposed to know everything, except the secret key. The security of a system relies only on the secrecy of the key but never on the privacy of the design.

This principle is also used in steganography and steganalysis. The embedding and extraction functions, and the distributions of all random variables, are all known to Eve.

1.4.2 *Cost based steganography*

In practice, Alice would like to achieve security while embedding a message. So it means respecting a condition on entropy as explained in section 1.3 while minimizing the detectability. Minimizing the KL divergence introduced in section 1.3.2 is ideal but difficult in practice; instead, Alice defines a distortion function and minimizes it while embedding afterwards.

The following mapping D typically measures the distortion:

$$\begin{aligned} D: \quad \mathcal{X}, \mathcal{Y} \quad &\to \mathbb{R}^+ \cup \{+\infty\} \\ \mathbf{x}, \mathbf{y} \quad &\mapsto D(\mathbf{x}, \mathbf{y}). \end{aligned} \tag{1.4.4}$$

We can cite for example the family of distortion introduced in [14], for $\mathbf{x} = (x_i)_{i\in[1,n]} \in \mathcal{X}, \mathbf{y} = (y_i)_{i\in[1,n]} \in \mathcal{Y}$:

$$D_\gamma(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|^\gamma, \tag{1.4.5}$$

which is called L_1 norm for $\gamma = 1$ and *energy of embedding changes* for $\gamma = 2$. Another distortion could also be:

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i}^{n} [x_i \neq y_i], \tag{1.4.6}$$

which counts the number of embedding changes.

Those distortions functions are additive, i.e., the impact of modification of each pixel can be obtained as a sum of individual impact. We give the following general definition:

Definition 1.4.1 (Additive distortion)**.** *A distortion is additive if it*

has the following property:

$$\begin{aligned} D(\mathbf{x},\mathbf{y}) &= \sum_{i}^{n} D(\mathbf{x},(x_0,\dots,x_{i-1},y_i,x_{i+1},\dots,x_n)) \\ &= \sum_{i}^{n} D(\mathbf{x},\mathbf{x}+\mathbf{e_i}(y_i-x_i)), \end{aligned} \tag{1.4.7}$$

where $\mathbf{e_i}$ *is an unitary vector of length* n *contained all* 0 *except a* 1 *at index* i*.*

The steganographic *noise* or *signal* is defined by the embedding modifications made to the cover, so $\mathbf{y}-\mathbf{x}$. This vector is often denoted $\mathbf{b}=\mathbf{y}-\mathbf{x}$, and its components b_i. In the case of LSB replacement, $\mathbf{b}\in\{-1,0,1\}^n$.

The additivity hypothesis is widely used in steganography because of its simplicity. It is a heavy assumption, as it neglects the obvious dependence of modifications between pixels on the detectability. Working under this assumption makes the definition of the distortion equivalent to the definition of a set of *costs of modification* where each component evaluates the impact of changing each pixel at index i by adding some value b:

$$\rho_i^b = D(\mathbf{x},\mathbf{x}+b\mathbf{e_i}), \tag{1.4.8}$$

such as the distortion can we rewritten as $D(\mathbf{x},\mathbf{x}+\mathbf{b})=\sum_i^n \rho_i^{b_i}$, or $D(\mathbf{x},\mathbf{y})=\sum_i^n \rho_i^{y_i-x_i}$.

Once the distortion is defined, it is up to Alice to embed a message according to it. We will see in the next section that Alice can have two different behaviours.

1.4.3 *Steganography and steganalysis metrics*

Paper [19] introduces two kinds of senders. The first one (DLS) maximizes the message length for a given distortion applied to the cover; the second (PLS) is the inverse of the DLS.

Definition 1.4.2 (DLS)**.** *For a given cover* $\mathbf{x}\in\mathcal{X}$*, the Distortion-Limited Sender (DLS) attempts to find a distribution* π *on* $\mathcal{Y}(\mathbf{x})$ *that has the highest entropy and whose expected embedding distortion does not exceed a given* D_ϵ*:*

$$\underset{\pi}{\text{maximize}} \quad H(\pi) = -\sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \pi(\mathbf{y})\log\pi(\mathbf{y}) \tag{1.4.9}$$

$$\textit{subject to} \quad \mathbb{E}_\pi[D] = \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \pi(\mathbf{y})D(\mathbf{x},\mathbf{y}) = D_\epsilon. \tag{1.4.10}$$

Alternatively, in practice, it may be more meaningful to consider the following complementary task, which is to embed a given payload with minimal possible distortion:

Definition 1.4.3 (PLS)**.** *The Payload-Limited Sender (PLS) attempts to find a distribution* π *that communicates a required payload* m*while minimizing the distortion:*

$$\underset{\pi}{\text{minimize}} \quad \mathbb{E}_\pi[D] = \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \pi(\mathbf{y})D(\mathbf{x},\mathbf{y}) \tag{1.4.11}$$

$$\textit{subject to } H(\pi) = m. \tag{1.4.12}$$

A theoretical result exists about the probability distribution π achieving the maximization or the minimization.

1.4.4 *Theoretical bound*

The PLS and DLS are constrained optimization problems. They can be theoretically solved using Lagrange multipliers (proof in [14]). The following theorem gives the probability distribution π, and the minimum average distortion:

Theorem 1.4.4 (Optimal distribution w.r.t. distortion)**.** *For a given cover* $\mathbf{x}$ *and the set of reachable stegos* $\mathcal{Y}(\mathbf{x})$*, the solutions of the PLS and DSL problem are both reached for the following distribution:*

$$\pi^\star(\mathbf{y}) = \frac{1}{A} e^{-\lambda D(\mathbf{x},\mathbf{y})}, \tag{1.4.13}$$

where A *is the scaling factor :*

$$A = \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} e^{-\lambda D(\mathbf{x},\mathbf{y})}, \tag{1.4.14}$$

and where λ *is determined from either from the entropy constraint, either by the distortion constraint, depending on the sender:*

$$\begin{cases} \displaystyle\sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \pi^\star(\mathbf{y}) D(\mathbf{x},\mathbf{y}) = D_\epsilon & \text{if DLS} \\ \displaystyle -\sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \pi^\star(\mathbf{y}) \log_2 \pi^\star(\mathbf{y}) = m & \text{if PLS} \end{cases}. \tag{1.4.15}$$

With m *or* D_ϵ *in the feasibility region of their corresponding constraints, the value of* λ *is unique*[4]*. This follows because both the expected distortion and the entropy are monotonically decreasing in* λ*.*

[4] In general, finding a solution for λ is achieved by binary search as the functions are monotonous.

As explained in the previous section 1.4.3, the distortion is often considered as additive. In that case, the optimal distribution $\pi^\star$ of the Theorem 1.4.4 can be rewritten[5] as a product of marginal probabilities of changing the individual pixels:

$$\pi^\star(\mathbf{y}) = \prod_i^n \frac{e^{-\lambda\rho_i^{y_i - x_i}}}{\sum_{b\in\mathbb{F}_q} e^{-\lambda\rho_i^b}} = \prod_i^n \pi_i^\star(y_i - x_i). \tag{1.4.16}$$

The definition of the marginal probability given by

$$\pi_i^\star(b) = \frac{e^{-\lambda\rho_i^b}}{\sum_{b'\in\mathbb{F}_q} e^{-\lambda\rho_i^{b'}}} \tag{1.4.17}$$

is very important and is widely used in the next sections. For example, if the modification b are in $\{-1, 0, +1\}$, then

$$\pi_i^\star(-1) = \frac{e^{-\lambda\rho_i^{-1}}}{e^{-\lambda\rho_i^{-1}} + e^{-\lambda\rho_i^{0}} + e^{-\lambda\rho_i^{+1}}}.$$

.

Given that theoretical bound, the efficiency of a coding function Emb can be evaluated w.r.t the distortion function D by comparing $\mathbb{E}[D]$ to $\mathbb{E}_{\mathbf{y}\sim\pi^\star}[D(\mathbf{x},\mathbf{y})]$, as the probability $\pi^\star$ is given in Theorem 1.4.4. If we observe that the embedding function gives an average distortion equal to the theoretical bound, we know that the embedding function is optimal, and there cannot exist another scheme that gives a lower

[5]
$$\begin{aligned} A &= \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} e^{-\lambda D(\mathbf{x},\mathbf{y})} \\ &= \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} e^{-\lambda \sum_i^n \rho_i^{y_i - x_i}} \\ &= \sum_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \prod_i^n e^{-\lambda\rho_i^{y_i - x_i}} \\ &= \prod_i^n \sum_{b\in\mathbb{F}_q} e^{-\lambda\rho_i^b}, \end{aligned}$$

so that

$$\begin{aligned} \pi^\star(\mathbf{y}) &= \frac{1}{A} \prod_i^n e^{-\lambda\rho_i^{y_i - x_i}} \\ &= \prod_i^n \frac{e^{-\lambda\rho_i^{y_i - x_i}}}{\sum_{b\in\mathbb{F}_q} e^{-\lambda\rho_i^b}}. \end{aligned}$$

distortion. For example, Figure 1.13 compares the average distortion per pixel of two previously designed coding functions (LSB replacement and matrix embedding using Hamming codes) with the optimal one, with binary changes and L_1 distortion. It is interesting to see that when the message length is smaller than the number of pixels, LSB replacement is suboptimal (we already observed this in section 1.2.3), but we see that matrix embedding using Hamming codes is also suboptimal.

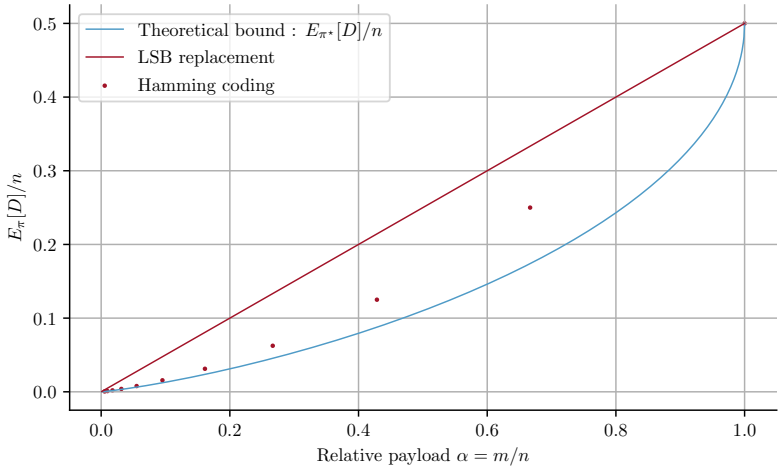


Figure 1.13: Comparison of the theoretical minimal average distortion given by Theorem 1.4.4, compared to average distortion given by LSB replacement or matrix embedding using Hamming coding, in the context of $\mathbb{F}_q = \mathbb{F}_2 = \{0, 1\}$ and where the distortion is defined by the L_1 norm. The x-axis is the relative payload α which is equal to the number of bits of message to embed per pixel. The y-axis is the average distortion per pixel.

The theorem is important because it reveals the *separation principle* which consists in separating the image model (needed for calculating the distortion) and the coding algorithm used in a practical implementation. Under this separation, better steganography can be derived using (i) better coding or using (ii) a better image model. One crucial outcome is that, to study the effect of the image model on steganographic security, no coding algorithm is needed. Flippling each pixel i according to the probability distribution $\pi_i^\star$ can simulate the optimal coding.

1.4.5 *Embedding while reaching the optimal bound*

Syndrom Treillis Code [20] is introduced by Tomas Filler in 2011 and is a technique allowing to reach this theoretical bound for any additive distortion. It uses matrix embedding and linear coding, such that it can embed any message with linear complexity. The overall idea is shortly introduced, but the details of this embedding scheme will not be explained here as it requires a solid theoretical background in linear coding.

The cover object $\mathbf{x}$ is modified to $\mathbf{y}$ such that the message $\mathbf{m}$ is communicated as a syndrome of a parity check matrix $\mathbf{H} \in \Sigma^{m \times n}$, i.e.

$$\mathbf{Hy} = \mathbf{m}. \tag{1.4.18}$$

Since $m < n$, the solution of $\mathbf{Hy} = \mathbf{m}$ is not unique. This freedom

is used to select $\mathbf{y}$ which not only communicates the message but also minimizes distortion measured by the function $D(\mathbf{x}, \mathbf{y})$.

The STC algorithm proposes the Viterbi algorithm to solve the above problem introduced in Equation 1.4.18, provided that

- the distortion function is additive, i.e.

$$D(\mathbf{x}, \mathbf{y}) = \sum_{i}^{n} D(\mathbf{x}, \mathbf{x} + \mathbf{e_i}(y_i - x_i)) = \sum_{i}^{n} \rho_i^{y_i - x_i},$$

- $\mathbf{H}$ is a sparse block-diagonal matrix constructed by repeatedly placing sub-matrix $\tilde{\mathbf{H}} \in \Sigma^{h \times \frac{n}{m}}$ next to each other and shifted down by one, as shows Figure 1.14. The parameter h is known in convolutional codes as the constraint height, and its value affects the performance of the algorithm: larger h will find solutions with lower distortion, at higher computational cost.

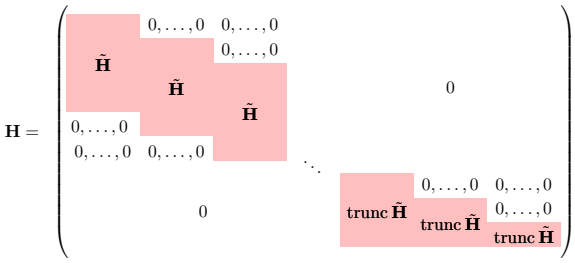


Figure 1.14: Block diagonal matrix used in Syndrome-Trellis Codes, as proposed in [20]. trunc $\tilde{\mathbf{H}}$ denotes $\tilde{\mathbf{H}}$ with bottom rows appropriately removed.

It is a game-changing paper as the steganography research field from this point is almost only focused on designing new distortion functions.

In this manuscript, the steganographic scheme will only consist, for a given cover image $\mathbf{x}$, in designing additive distortion functions characterized by the set of individual costs ρ_i^b. A steganalyzer tests the performance of this distortion function by analyzing stego images drawn from the optimal distribution, i.e. $\pi^\star$ which is directly computed from Equations 1.4.15 and 1.4.16 for a given payload m or distortion D_ϵ.

The next section 1.5 presents state-of-the-art distortion functions in chronological order, with illustrations in Appendix.

1.5 Practical distortion functions

The design of distortion functions is a vast subject of research. Researchers often use a trial and error method, with state-of-the-art steganalysis providing feedback. All following distortion functions are designed (or approximated) as additive and for ± 1 embedding.

1.5.1 *HUGO*

Before 2010, steganalysis methods were more developed than steganography. Presentation of the Highly Undetectable steGO [21] scheme made much noise as it was introduced during the BOSS (Break Our

Steganographic System) competition and was achieving excellent performance. It gave a hard time to contestants and made the competition very challenging. The idea is to adapt the cost directly from the SPAM features [22], which were used by the best steganalyzer at the time. Costs responsible for the detection of LSB matching were identified using Fisher Discriminant and were heuristically tuned.

1.5.2 *MG*

The Multivariate Gaussian model [23] comes in 2013 with a computation of costs made to decrease the KL divergence between the distribution of cover and stego images. To do so, cover images are modeled as a sequence of independent (but not necessarily identically distributed) quantized Gaussians, whose distribution is denoted by $p^{(i)}$. The distribution of stego pixels, which are also independant and denoted $q^{(i)}$, are derived in order to decrease $D_{\mathrm{KL}}\left(p^{(i)} \| q^{(i)}(\pi_i)\right)$. For small change rates, the KL divergence is well-approximated with its leading quadratic term:

$$\sum_{i=1}^{n} D_{\mathrm{KL}}\left(p^{(i)} \| q^{(i)}\left(\pi_i\right)\right) \approx \sum_{i=1}^{n} \frac{1}{2} \pi_i^2 I_i(0) \tag{1.5.1}$$

where $I_i(0)$ is the steganographic Fisher information (FI)[6]:

$$I_i(0) = \sum_j \frac{1}{p_j^{(i)}} \left(\frac{\mathrm{d}q_j^{(i)}\left(\pi_i\right)}{\mathrm{d}\pi_i}\bigg|_{\pi_i=0} \right)^2 . \tag{1.5.2}$$

[6] Fisher information is a quantity that frequently appears in the theoretical steganography and, in general, in signal detection and estimation. FI appears in the leading term of the Taylor expansion of the KL divergence so that zero KL divergence implies zero FI.

The optimal choice of π_i that minimizes the total KL divergence in Equation 1.5.1 subjects to the payload constraint in Equation 1.4.15. The problem is solvable by using the method of Lagrange multipliers. It gives an equation that must be solved numerically for each pixel i.

In order to theoretically embed a message while producing stegos according to the distribution defined by the previous equation, one can derive costs as follows:

$$\rho_i = \ln\left(1/\pi_i - 1\right). \tag{1.5.3}$$

1.5.3 *ASO*

The Adaptive Steganographic scheme by Oracle (ASO) [24] is also published in 2013 and is the first spatial steganographic scheme whose distortion function is directly computed from an oracle which is the steganalyzer. The detectors (the best at the time) are in the paper the Kodovsky's ensemble classifiers noted f_k.

The costs are computed like:

$$\rho_i = \sum_k \left[f_k(\mathbf{x} + \mathbf{e_i}) - f_k(\mathbf{x})\right]. \tag{1.5.4}$$

This scheme relies on a new philosophy. The costs are directly made to escape a specific steganalyst, whereas previous schemes are indirect.

1.5.4 *UNIWARD*

The embedding distortion of UNIWARD [25] in 2014 comes with a general formula for both spatial and JPEG domain, and it is computed as a sum of relative changes of coefficients in a directional Daubechies wavelet filter bank decomposition of the cover image. The specificity of filters forces the embedding changes to parts of the cover object that do not resonate with the filter, such as textures or noisy regions while avoiding smooth regions or clean edges.

For $\mathbf{x}, \mathbf{y}$ two images in the spatial domain, the S-UNIWARD distortion function between the two is defined as:

$$D(\mathbf{x},\mathbf{y}) \triangleq \sum_{k=1}^{3}\sum_{i=1}^{n} \frac{\left|W_i^{(k)}(\mathbf{x}) - W_i^{(k)}(\mathbf{y})\right|}{\sigma + \left|W_i^{(k)}(\mathbf{x})\right|}, \tag{1.5.5}$$

where $\sigma > 0$ is a constant stabilizing the numerical calculations. For images in the JPEG domain, the same formula is applied but by first decompressing the image in the spatial domain by applying the inverse JPEG transform $J^{-1} : D(J^{-1}(\mathbf{x})), J^{-1}(\mathbf{y}))$.

$W^{(k)}(\cdot)$ is the operation of convolution by $K^{(k)}$ which are filters from the Daubechies wavelet filter bank build from 8 Daubechie wavelet 1D $\mathbf{h}$ low-pass and $\mathbf{g}$ high-pass (plotted on Figure 1.15). The three 16×16 filters $K^{(1)} = \mathbf{h}\cdot\mathbf{g}^\top, K^{(2)} = \mathbf{g}\cdot\mathbf{h}^\top, K^{(3)} = \mathbf{g}\cdot\mathbf{g}^\top$ computed from those vectors are plotted on Figure 1.16

For simplification, an additive approximation is made such as:

$$\rho_i^b(\mathbf{x}) = D(\mathbf{x}, \mathbf{x} + \mathbf{e_i}b). \tag{1.5.6}$$

For ±1 embedding, $b \in \{-1, 0, 1\}$, so $\rho_i^0 = 0$ and $\rho_i^{-1} = \rho_i^{+1}$. Hence the distortion can be described by only the cost of modification ρ_i, as well as the optimal probability of change π_i for pixel i.

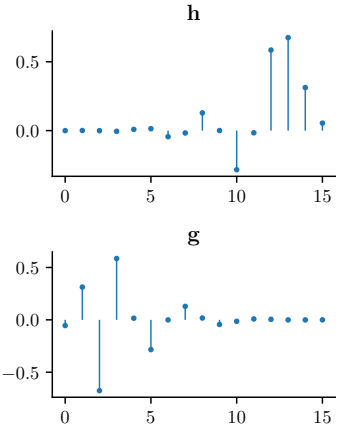


Figure 1.15: Daubechies 8 wavelet decomposition vectors: $\mathbf{h}$ low-pass and $\mathbf{g}$ high-pass.

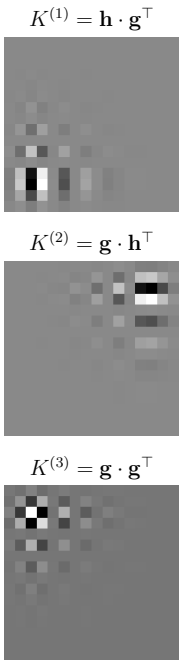


Figure 1.16: Daubechies wavelet filter bank $K^{(1)}, K^{(2)}, K^{(3)}$ build from $\mathbf{h}$ and $\mathbf{g}$.

1.5.5 *SI-UNIWARD*

In the same paper presenting UNIWARD [25], another cost function is designed depending moreover on the raw DCT coefficient obtained from the precover $\mathbf{p}$, called *side information*. If $\mathbf{x}$ is a JPEG, there exists an original precover $\mathbf{p}$ such as $\mathbf{x}$ was obtained by JPEG compression of $\mathbf{p}$, i.e. through DCT transform (by blocks of 8×8), quantization and rounding. $\mathbf{u}$ is defined as $J(\mathbf{p})$ ie the (non-rounded) DCT coefficients of the uncompressed precover $\mathbf{p}$. In this scenario, Alice has access to different objects than usual because *precover* is the raw data given by the photographic sensors of the camera.

The idea of the side-informed (SI) cost function is to take into account the rounding error, which is the absolute difference between $\mathbf{x}$ and the image obtained from $\mathbf{p}$ with JPEG compression but without the last rounding step.

The SI-UNIWARD cost function is defined by:

$$D^{(\mathrm{SI})}(\mathbf{x},\mathbf{y}) \triangleq D\left(\mathbf{p}, J^{-1}(\mathbf{y})\right) - D\left(\mathbf{p}, J^{-1}(\mathbf{x})\right). \tag{1.5.7}$$

This approach favours changes on the elements whose rounding errors are close to $\pm 1/2$ because they are the coefficients the more naturally prone to be modified during compression if they are changed by a small perturbation.

The additive approximation proposed in [25] is corrected in paper [26] and leads to the following definition

$$\rho_i^{b^{(\mathrm{SI})}}(\mathbf{x}) = (1 - 2\,|e_i|)\,\rho_i^b(\mathbf{p}) \tag{1.5.8}$$

where $\mathbf{e}$ is equal to the rounding error ie $e_i = u_i - x_i$.

1.5.6 *HILL*

The HILL (HIgh-pass, Low-pass, and Low-pass) [27] scheme from 2014 is outstanding by its simplicity and efficiency. It aims at joining low costs to textured regions and high costs to pixels belonging to homogeneous areas. Charaterization of texture is achevied by a 3×3 high-pass filter H and by two low-pass filters L_1 of size 3×3 and L_2 of size 15×15. The symmetric costs are obtained by the following formula:

$$\rho = \frac{1}{|\mathbf{x} \star H| \star L_1} \star L_2. \tag{1.5.9}$$

$H = \begin{pmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{pmatrix}$. L_1 and L_2 are constant matrices, of value $1/9$ and $1/225$.

The design of the filters required much investigation. The authors tried multiple versions of the low and high pass filters.

1.5.7 *Synchronization of modifications*

In order to reduce the detectability of an insertion scheme, a possible strategy is to synchronize the modifications. If in the spatial domain, a pixel is modified by a $(+1)$ operation, it may be interesting to favor a $(+1)$ insertion rather than a $(+0)$ or (-1) one on a neighboring pixel, in order to favor the appearance of a stego signal whose variations are similar to those of the image. Such synchronization may be introduced by embedding the message in chunks then modifying the distortion according to the embedding *sequentially*.

The use of lattices is made to embed successively pieces of the message, where the natural sampling grid of coefficients encoding the image is split into an ensemble of disjoint grids, as shown in Figure 1.17. A piece of the message is hidden in the first lattice (for example Λ_1) by minimizing the additive distortion; then, the distortion is calculated for the second lattice given the modifications made to the first lattice.

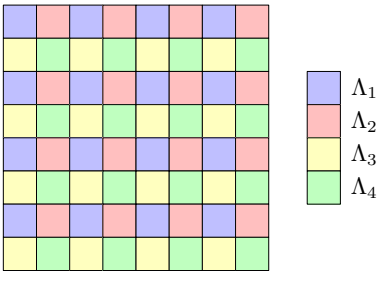


Figure 1.17: Example of repartition of image coefficients into four latttices $\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4$.

Clustering Modification Direction (CMD [28]), is an example of such synchronization scheme. It can be used with the cost proposed by HILL in section 1.5.6. For example, for cost of modification +1, and μ_i is the average of modification made in adjacent coefficients of coefficient i, the new cost $\rho_i'^{+1}$ is given by:

$$\rho_i'^{+1} = \begin{cases} \frac{1}{9}\rho_i^{+1} & \text{if } \mu_i > 0, \\ \rho_i^{+1} & \text{if } \mu_i \leq 0. \end{cases} \tag{1.5.10}$$

It favours a +1 modification of a pixel (by lowering its cost by an arbitrary factor 1/9) if the neighbourhood has been modified by +1 changes in average.

1.5.8 *UERD*

Computation of costs proposed by UERD [29] (Uniform Embedding RevisiteD) associates in 2015 each quantized DCT coeffcient of the JPEG image depending on the quantization step q_i and to the energy of sourrounding blocks. The energy $D_{\mathcal{B}} = \sum_{i \in \mathcal{B}} q_i |x_i|$ of the block $\mathcal{B}$ is defined as the sum of coefficients mulitplied with the quantization step. The symmetric cost of +1 or −1 are defined as:

$$\rho_i = \frac{q_i}{D_{\mathcal{B}} + 0.25 \sum_{k=1}^{8} D_{\mathcal{B}^k}}, \tag{1.5.11}$$

where the sum is over the 8 blocks surrounding $\mathcal{B}$ where $\mathcal{B}$ is the block containing pixel i.

1.5.9 *MiPOD*

MiPOD [30], as Minimizing the Power of Optimal Detector, published in 2016, is close to MG [23] presented in section 1.5.2 because the steganographic scheme works under the assumption of cover being modeled by a Gaussian distribution. Its difference is that MG minimizes the KL divergence between the cover and stego distributions in the asymptotic limit of a small payload, while MiPOD minimizes the power of the most powerful detector, which is achieved without the additional assumption of a small payload. The optimal detector is obtained via the optimal Likelihood-Ratio Test (LRT).

1.5.10 *ADV-EMB*

Finally, ADVersarial EMBedding [31], also called ADV-EMB comes in 2019 and its philosophy is close to ASO's (see section 1.5.3) because it aims at updating a portion of the costs depending on the output of a detector. When the steganalysis task is produced by a differentiable classifier f, the costs of +1 or −1 modifications are updated according to:

$$\hat{\rho}_i^j = \rho_i^j \alpha^{j \operatorname{sign}\left(\frac{\partial f(\mathbf{s})}{\partial s_i}\right)}, \tag{1.5.12}$$

with α set to 2. Section 3.1 details the exact scheme of ADV-EMB profoundly because it is an attack that we reproduced in our contribution. Now that we introduced most steganographic schemes, it is time to introduce the main steganalysis methods.

1.6 Practical steganalysis methods

Steganalysis tools have always been constructed in order to detect some already existing steganographic schemes, as we can see in Figure 1.20 and 1.21. They were for a long time designed heuristically

by custom features extraction followed by co-occurrence or histogram construction and finally a classical classification method, as pictured in Figure 1.18. The classification method is often reached with SVM, which is a machine learning algorithm. It is not explained here, and we will focus only on features extraction.

Non-automatic methods rely on finding the significant characteristics in an image to discriminate the covers from the stegos. This empirical step is called *feature extraction*. Because stegos are often obtained by a slight modification to the cover, it is widely assumed that the relevant features can be computed by modelization of *noise residuals* of cover images. It is the weak signal obtained by linear and non-linear filters applied in the camera (for example during demosaicing operation).

The computation of residuals depends on the format of the image (spatial or JPEG), and the methods are presented in the following subsections. The most recent steganalysis do not require feature extraction, but instead an *architecture* design of the CNN, as chapter 2 shows.

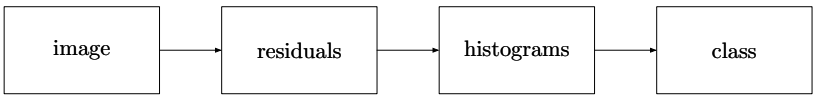


Figure 1.18: General steps of steganalysis. The last step is classification, outputing if the image is in the cover or stego class.

1.6.1 *Rich models*

Rich models [32] designed in 2012 is a complex model consisting of multiple submodels, each capturing slightly different embedding artefacts. SRM and JRM (for spatial or JPEG) aim at extracting the features from the noise residuals computed using high-pass filters because the stego signal is not contained in the content of the image but inside the noise. Then, one or several co-occurrence matrices of neighbouring samples from the truncated and quantized residuals are computed, such as finally a classifier is trained given those matrices.

1.6.2 *DCTR*

The DCTR [33] features (Discrete Cosine Transform Residual), published in 2014, use 64 kernels of the discrete cosine transform. The decompressed JPEG image is convoluted with each DCT kernel firstly, and then the first-order statistics of quantized noise residuals are obtained by subsampling residual images. The DCTR features can achieve better detection performance while preserving relatively low feature dimensions.

1.6.3 *GFR*

The GFR [34] methodology (Gabor Filter Rich models, 2015) is inspired from the previous DCTR, but it differs from the filters used.

Instead of the 64 DCT kernels, it creates steganalysis features using 2D Gabor filters with different scales and orientations.

A Gabor filter, named after Dennis Gabor, is a linear filter regularly used for texture analysis. It analyzes whether there is any specific frequency content in the image in specific directions in a localized region around the point or region of analysis. An example of a Gabor filter is in Figure 1.19.

As for DCTR, after the steganalysis features are extracted from the image, the classification operation is applied.

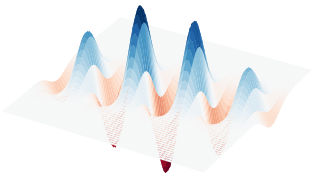


Figure 1.19: Example of 2D Gabor filter

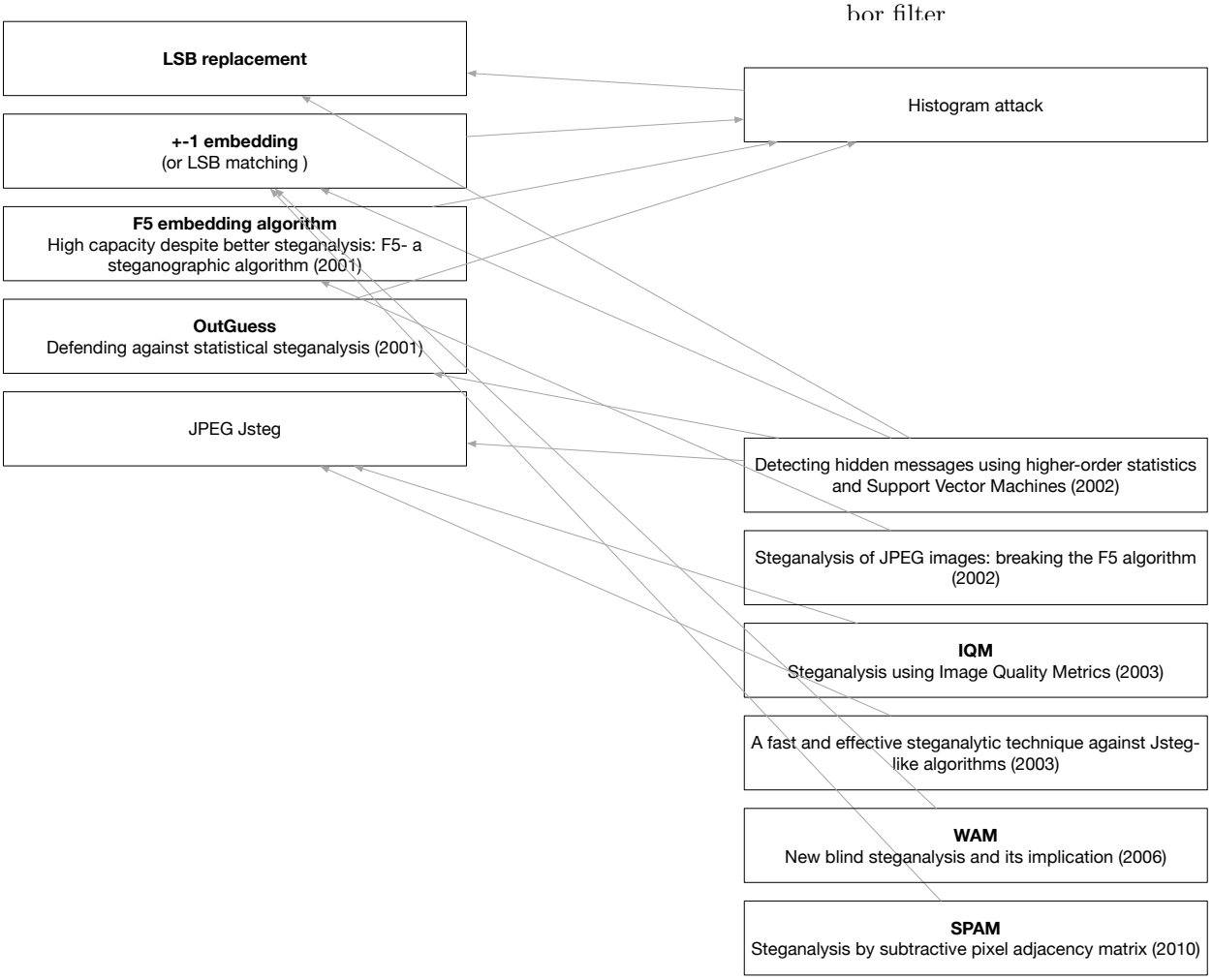


Figure 1.20: Historical point of view of the mouse-and-cat game (up to bottom) of steganography (left column) advancees and steganalysis (right column) (part 1/2).

1.7 Conclusions of chapter

We presented in this chapter the essential knowledge to understand how works steganography and steganalysis with digital images.

When Alice transmits a message $\mathbf{m}$ to Bob by embedding it into a natural image $\mathbf{x}$, she modifies $\mathbf{x}$ to obtain the stego $\mathbf{y} = \mathbf{x} + \mathbf{b}$ by adding the stego signal $\mathbf{b}$. Alice and Bob previously agree about the coding/decoding method and a secret key. Commonly, Alice designs

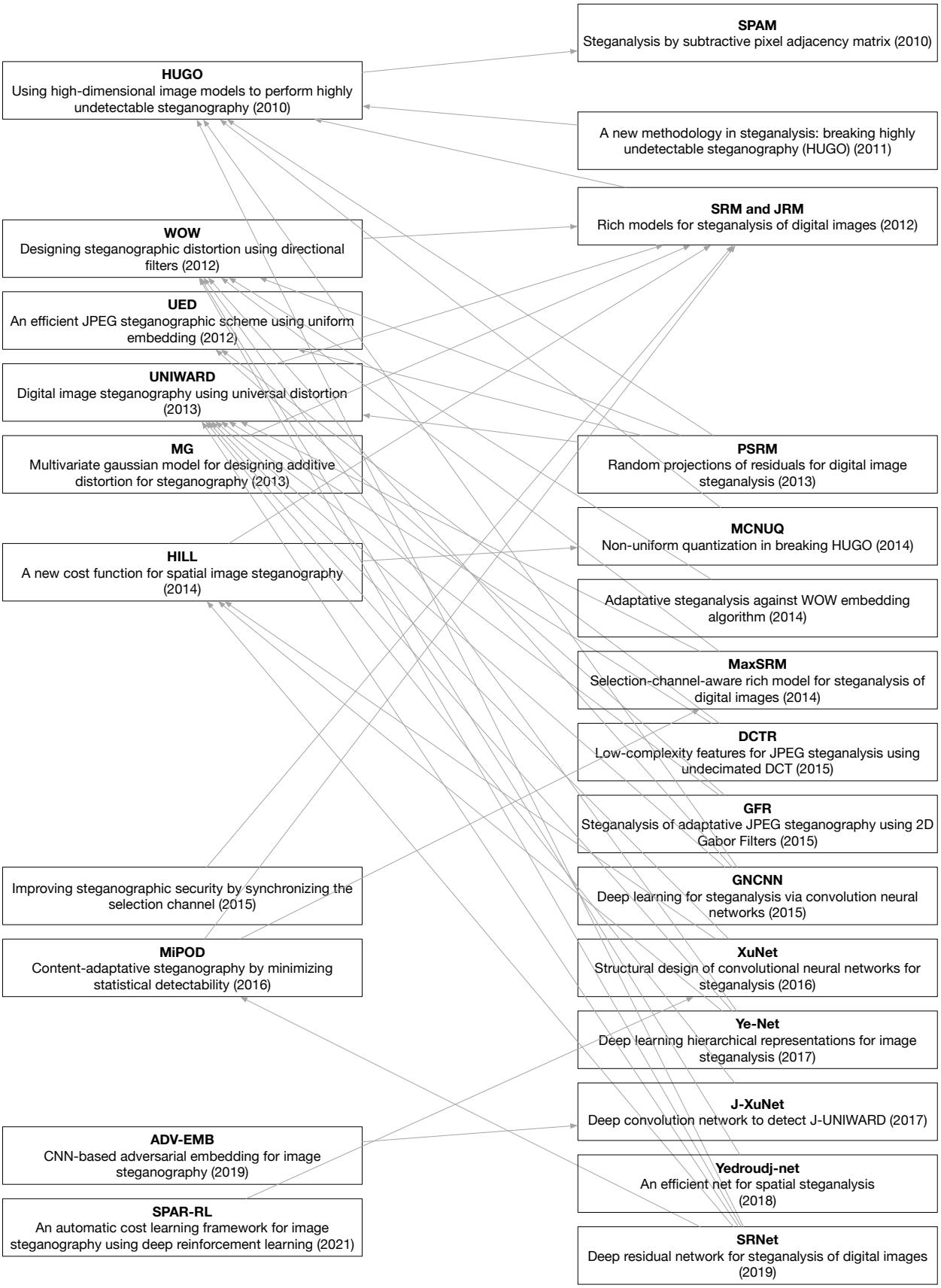


Figure 1.21: Historical point of view of the mouse-and-cat game. (part 2/2).

$\mathbf{b}$ such as $\mathbf{Hy} = \mathbf{m}$, where the matrix $\mathbf{H}$ is generated with the shared key.

Alice tries to adapt the embedding to the image's content to respect the natural aspect of images. She tries to determine what features to preserve and designs a distortion function assigning a cost of modification to each image coefficient (often, the distortion function is assumed additive). Then Alice uses a coding method to find $\mathbf{b}$, such as Bob decodes the correct message while minimizing the distortion function (the value of the minimum depends on the length of the message). In practice, we simulate the embedding and draw $\mathbf{b}$ according to a probability distribution given by the theoretical lower bound of the distortion limited by the length of the message.

We studied many distortion functions, from the simplest one, equals to the number of modifications made to the cover, to more elaborated ones like HILL analyzing the smooth and textured regions to attribute respectively high or low costs.

Then we explained how to measure the security of a steganographic scheme. Cachin defines it as the KL divergence between the distribution of cover and stego images. However, it is non-practical as the distribution of natural covers is not available in general. In practice, the security is measured by the performance of steganalysis tools, under Kerckhoffs' principle, stating that Eve knows the embedding scheme, except the secret key.

Therefore, we studied the classical method of steganalysis. It starts with an extraction feature part, followed by the classification phase. The more basic features rely on the distribution of variations between adjacent coefficients in an image, and a threshold can do the classification. More complex features rely on noise extraction and histograms computation.

Today, the most performant tools are achieved by deep learning models, where the features extraction phase is automated. It is the subject of the next chapter.

Basics of Deep Learning for steganalysis

2

2.1 Machine learning

In nature, as our measure instruments become more and more sophisticated and precise, the phenomena happening around us appear to be predictable or intrinsically bound. For example, the trajectory of planets seem to follow a ceaseless elliptical movement, or the objects seem always to fall with the same acceleration. As soon as we were able to measure those observations, also called *data*, humans had the abstraction ability to describe those patterns with very general formulas made of a combination of symbols. Those theories, also called models, are made such as it confirms their observations. It also has the powerful ambition to predict the results of future situations. In other words, they are made to be *general*.

Machine learning is a category of computer algorithms that aims at producing *models* automatically through experience and by the use of data. From the so-called training data, we expect the algorithm to be able to *generalize* and to perform well on new data: if it does so, we say that the program learns.

Those tools are mainly used when we do not manage to conceive a straightforward (typically rule-based) algorithm, often because of the high dimensionality of the problem. In this case, we expect machine learning to find the model that humans did not manage to find because of its complexity. It is why it is used in steganalysis.

Because the program is on a computer, we must first translate the task mathematically. Then, for some input data x, we want to learn a *model* f such that the output $y = f(x)$ fulfills the desired task. The following section illustrates an example of designing such function f, which fits natural observations.

2.1.1 *Purpose of machine learning with an example*

For example, let us imagine that a scientist observes that some measurable quantity x seems correlated to another quantity y and wishes to find the law bounding the two. They collect 20 examples of values (x_i, y_i) and plots them on a graph on Figure 2.1. Note that measurement always comes with some noise.

Finding the model f will allow him to know the supposed value of $y = f(x)$ associated with a new value of x that he/she did not measure, or vice versa.

To do so, he needs to choose some hypotheses about the model. For

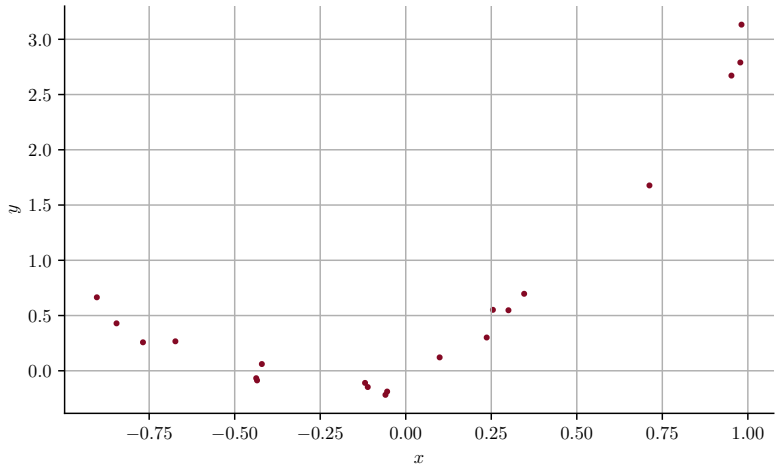


Figure 2.1: Set of experimental points $\{(x_i, y_i)\}$ measured by a scientist.

example, it could be that a quadratic equations bonds the variables x and y, i.e. it might exists some real numbers a, b, c such as $f(x) = ax^2 + bx + c = y$.

Under this assumption, we can search the three unknowns a, b, c such as $f(x_i) = y_i \forall i$. To do so, we could rely on a *loss function* L evaluating the model f by making the loss minimal when f generalizes well, and higher when f does not. This gives us a mathematical function L to minimize, because it reaches its minimum for a satisfying model. We will see in section 2.3 that minimization can be achieved via an optimization algorithm. We often require the loss function to be differentiable in order to simplify the optimization process.

Once the loss is chosen, we can finally apply an optimization algorithm to update typically iteratively the parameters of the model a, b, c. Figure 3.9 shows three different steps of the optimization process (it is a multistep algorithm).

The loss of the model f could be for example the sum of the squares of distances between the real value y_i and the predicted values $f(x_i)$ (also called mean squared error or MSE) : $\text{MSE}_f = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2$

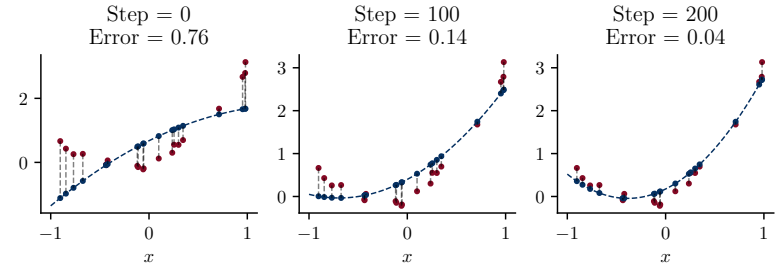


Figure 2.2: Evolution of the model f defined as $f(x) = ax^2 + bx + c$ during the optimization of the parameters a, b, c for minimizing the loss.

Fitting the data is not a goal itself because fitting the data perfectly without error can be made by interpolating the data with a lagrangian polynomial as shown the right plot in Figure 2.3. This phenomenon is called *overfitting*, as the "too good" fitting of the model alters prediction on unseen samples. It shows a compromise between error on available samples and the model's error evaluated on unseen samples.

In order to avoid overfitting, we could promote sparsity by choosing

For a given set of points $\{(x_i, y_i)\}$ with no two x_i values equal, the Lagrange polynomial ℓ is the polynomial of lowest degree that assumes at each value x_i the corresponding value y_i so that the functions coincide at each point. ℓ is given by $\ell_j(x_i) = \prod_{\substack{m=0 \\ m \neq j}}^{k} \frac{x_i - x_m}{x_j - x_m}$

a model with as few parameters as possible. It could be achieved via *regularization* as well[1]. Figure 2.3 shows the effect of regularisation by a parsimonious choice of the model: the left plot is where a model has not enough parameters (2), and the right plot for too many parameters (15). They produce respectively *underfitting* and *overfitting* and both are bad models for unseen examples.

[1] Adding a *regularization loss* can apply regularization to a model. Often, it penalizes parameters with extreme values.

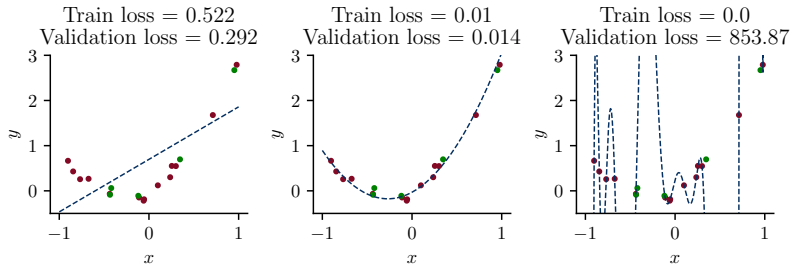


Figure 2.3: Value of the loss on the training (red points) or validation set (green points) for different models trained each on the same train set: (left) $y_i = ax_i + b$, (middle) $y_i = ax_i^2 + bx_i + c$ and (right) the lagrangian polynomial of degree 15 interpolating the training data.

On a more general note, machine learning consists in the steps inside the *while loop* of the following algorithm:

Data: Data split in train, validation and test set
Result: A satisfying model fitting test data
while *Model does not generalize well on the validation data* **do**
1. Design the model which could suit the situation, i.e. the parameters θ of some function f and how they are linked together to produce $y = f_\theta(x)$;
2. Choose a loss function that penalizes a model which does not fit the train data and equivalently favours a model which fits the train data ;
3. Apply an optimization algorithm whose aim is to minimize the loss defined before ;
4. Evaluate the loss of the optimized model on the validation data.

end

Algorithm 1: Loop of a machine learning model designer

Each of the three first steps 1-3 are the subjects of the three next sections. Section 2.2 shows the range of specific models of convolutional neural networks. Section 2.2.9 is for the design of a loss function. Finally, section 2.3 explains what an optimization problem is and how to solve it.

2.2 Convolution Neural Network (CNN)

Deep learning is a category of machine learning models inspired by the human brain's architecture. A deep learning model, also called Artificial Neural Network (ANN), is a function f composed of several *base* operations (parametrized by parameters θ) presented from section 2.2.2 to 2.2.6. When an input (often multidimensional) is fed to the model, it is typically subject to several successive transformations. As shown in Figure 2.4, the result of the transformations at different levels is called *layer*. The layer might be described as *input*, *hidden* or

output layer depending on its position in the architecture sequence.

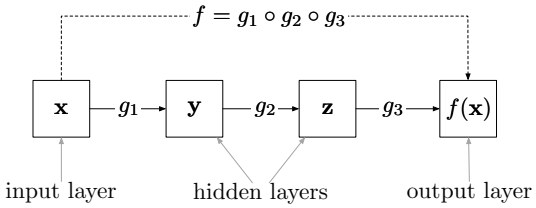


Figure 2.4: Representation of the 4-layers neural network. The input $\mathbf{x}$ can be mutlidimensional, like all other layers $\mathbf{y}, \mathbf{z}, f(\mathbf{x})$.

Because the layers might be multidimensional vectors, they may be composed of several elements. Each element in a layer is called *neuron*. We show in Figure 2.5 the classical representation of neural networks where a circle represents each neuron.

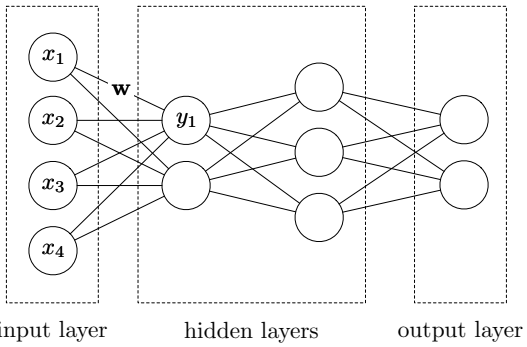


Figure 2.5: Classical representation of neural networks. The line between layers significate operations between neurons. Those operations are parametrized by learnable parameters.

The adjective *deep* refers to the multiple stacked operations which compose the model. The more there are layers, the deeper the model is. The *base* functions are often differentiable because it simplifies the optimization process as shown in section 2.3.

The whole structure of a neural network is often called a *graph*, as we can draw each variable in nodes and draw lines labelled by a function between them, like in Figure 2.5. In the case of images, a particular class of ANN exists, which are called convolution neural network (CNN). It differs by the type of operations executed in the graph, which are introduced from 2.2.1 to 2.2.9.

2.2.1 *Computational graph*

Neural network architectures can be seen as directed graphs whose variables are in the nodes. The nodes are connected with a line when a variable is obtained from another with a basic operation. The graph contains all the *symbolic path* from the input data x to the output $\hat{y}$. We can execute a *forward propagation* with the current value of variables contained in the nodes if we *feed* some data in the input.

We will see in section 2.3.3 that the graph is used as well to do a *backward propagation* during the learning process.

As explained in Algorithm 1, the first step of machine learning is designing a model f_θ such that it exists a vector of parameters $\theta^\star$ for which $f_{\theta^\star}$ fits the test data. For example, on Figure 2.6 is shown the graph of the model used in section 2.1.1.

The symbolic graph and all its symbolic operations are called *hyperparameters*. They are designed during the loop of Algorithm 1. The

variables contained in the nodes are often randomly initialized and are supposed to be optimized during step 3: they are called *parameters*.

Designing the graph of f_θ is essential, and it requires knowledge and experience as it may vary depending on the requested task. Hopefully, many research fields are interested in designing the best graphs, depending on the mission. When the input is images, a specific type of graphs provided with particular operations is used: they are called *Convolution Neural Networks*. They are trainable in practice (with not too many parameters) and adapted to the structure of images. They are built upon convolution operations which are denoted with the $\star$ symbol.

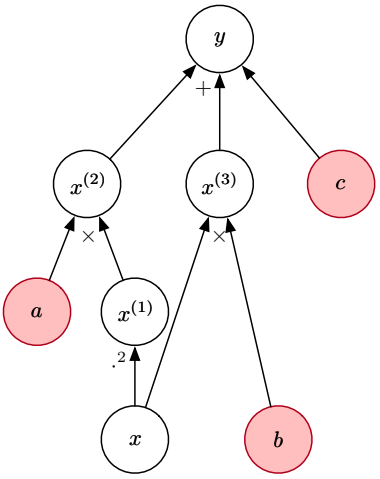


Figure 2.6: Example of the computational graph of the model $y = ax^2 + bx + c$. Red nodes are optimizable parameters. $x^{(i)}$ are intermediate expressions.

2.2.2 *Definition of a convolution*

The convolution operation was first very useful in signal processing. This operation is often called *filtering*, as it aims at selecting a type of pattern in the input signal. Because we are working with images represented with 2D discrete vectors, we give the following definition:

Definition 2.2.1 (Convolution for 2D discrete data)**.** *The convolution between image* $\mathbf{x} \in \mathcal{M}^{n_1,n_2}$ *and* $\mathbf{K} \in \mathcal{M}^{k_1,k_2}$ *is defined by the following matrix* $O \in \mathcal{M}^{m_1,m_2}$*:*

$$O[p,q] = (\mathbf{x} \star \mathbf{K})[p,q] = \sum_{i=p+1}^{p+k_1} \sum_{j=q+1}^{q+k_2} \mathbf{x}[i,j]\mathbf{K}[i-p, j-q]. \qquad (2.2.1)$$

The output size can be obtained from the size of $\mathbf{x}$ *and* $\mathbf{K}$*:* $m_1 = n_1 - k_1 + 1$ *and* $m_2 = n_2 - k_2 + 1$.

Usually, the size of the filter (k_1, k_2) is smaller than the input image size (n_1, n_2), so that we can see K as a small matrix superposed to the input images $\mathbf{x}$ (for a term to term multiplication then summation) sliding in both directions to obtain a matrix of size less or equal than the input image, as illustrated in Figure 2.7.

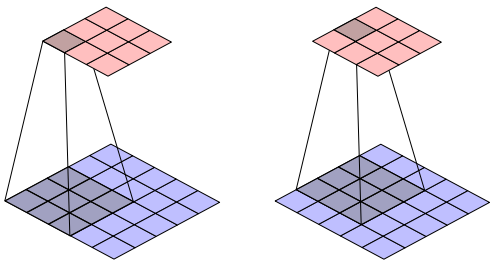


Figure 2.7: 3×3 filter (gray) sliding on 5×5 input image (blue) to produce 3×3 output (red).

We can design custom filters for a variety of tasks. For example, the following 3×3 kernel are (a) an averaging or low-pass filter, (b) a vertical edge detector and (c) a horizontal edge detector. Visualization of the output of those filters is shown in Figure 2.8.

$$\underset{\text{(a)}}{\begin{pmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{pmatrix}} \quad \underset{\text{(b)}}{\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}} \quad \underset{\text{(c)}}{\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}}$$

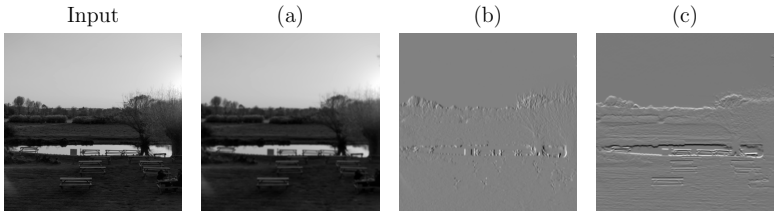


Figure 2.8: Input (left) convolved with filter (a), (b) or (c).

2.2.3 *Convolution in a CNN*

When a convolution operation appears in the architecture of a CNN, the filters (or kernels) are seen as parameters waiting to be optimized during the training phase. Indeed, we gave in the previous section 2.2.2 the example of easily solvable tasks as detecting edges. However, there exists more complex tasks such as detecting an animal in an image. The design of the filter is not straightforward; the purpose of deep learning is to find the correct parameters, which are the values of filters. In this way, the filters will be correctly designed to detect some patterns for a specific task. Values of the filters are also called *weights*.

In the context of a CNN, the coding of convolution operation requires another hyperparameter which is *striding*. Striding decreases the output size. It is a way to apply *subsampling* to the data. It is coded by a single integer (it could be coded with an integer for each dimension for input data). The stride controls the step size of the sliding. The default stride is 1. An example with value 2 is shown on Figure 2.9

Finally, we saw that 2D convolutions are well-adapted operations made to images for preserving the grid-like information. Their main advantage is to lower the number of parameters to optimize the architecture.

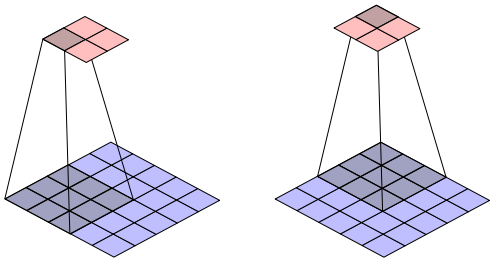


Figure 2.9: 3×3 filter (gray) sliding on 5×5 input image (blue) to produce 5×5 output (red) with stride of 2 and no padding.

2.2.4 *Pooling*

Pooling operations are used to reduce the dimension of the layers. Thus, it reduces the number of parameters to learn and the amount of computation performed in the model. The two most used pooling are *average pooling* and *maximum pooling*.

Average pooling is a convolution operation, with a constant filter whose sum equals 1. This filter is not a parameter; it does not change

during the training.

Maximum pooling looks like a convolution operation, but the sliding filter does not produce a term to term multiplication followed by summation; it does a maximum operation.

Usually, those operations are applied with a stride equal to the filter window size with no padding[2]. Figure 2.10 shows an example of such pooling.

[2] Padding consists in adding pixels as a frame around the image whose values depend on the kind of padding (zero padding fill the pixels with 0 for example). Padding is often used such that the size of the output equals the size of the input.

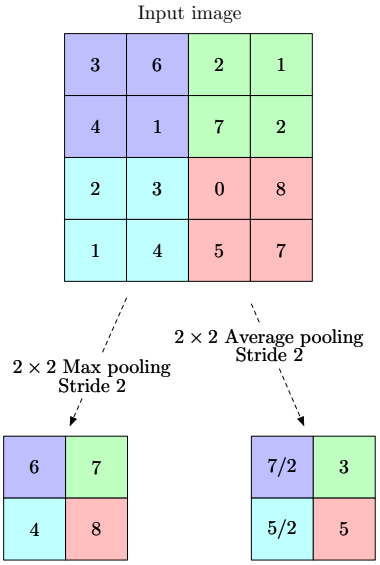


Figure 2.10: Effect of maximum pooling or average pooling applied an input image.

2.2.5 *Activation functions*

The *activation* operation introduces nonlinearities in a layer. If it were not used, the network would remain a linear function of its input, limiting the range of possible modelling. It is shown in [35] that using non-linear function is capital as it allows to learn the basic XOR operation.

There exists many activations functions, and historical the used ones are defined below:

$$\text{Step}(x) = \begin{cases} 0 \text{ if } x \leq 0, \\ 1 \text{ if } x > 0 \end{cases}, \tag{2.2.2}$$

$$\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}, \tag{2.2.3}$$

$$\tanh(x) = \frac{e^{-x} - e^{-x}}{e^{x} + e^{-x}}. \tag{2.2.4}$$

In modern neural networks, the default recommandation is to use the *rectified linear unit*, or ReLU [36], defined by

$$\text{ReLU}(x) = \max\{0, x\}. \tag{2.2.5}$$

All the defined functions are depicted in Figure 2.11. The definitions are extended for vector input by applying the function term-wise.

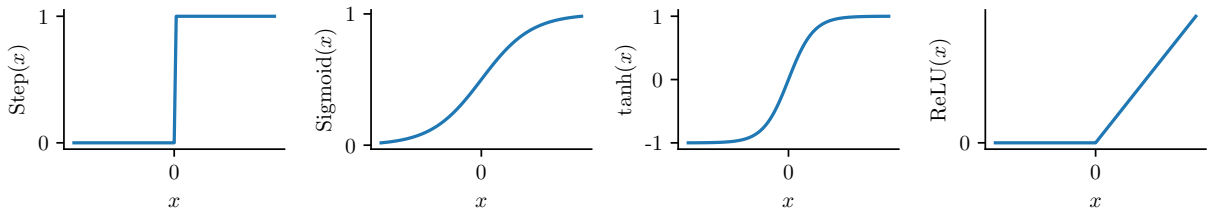


Figure 2.11: Activation functions defined from Equation 2.2.2 to 2.2.5.

The activation function is applied after a convolution operation or is the final computational step of a fully connected layer, defined in the next section.

2.2.6 *Fully connected layer*

After several convolutional and pooling layers, the final classification is done via fully connected layers, or also called *dense layer*. A linear

transformation connects all neurons in input of a fully connected layer to all neurons of its outputs. If $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^m$ are respectively the input and output vectors, we have

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{2.2.6}$$

where $\mathbf{M} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$. The operation $\mathbf{x} \mapsto \mathbf{W}\mathbf{x} + \mathbf{b}$ is at the core of the fully connected layer. The elements of $\mathbf{W}$ and $\mathbf{b}$ are optimized during training ; they are respectively called *weights* and *bias*.

Ideally, fully connected layers could be applied in each layer. But for images with a lot of features, this would make too many parameters so the learning would be too costly. It is why it is replaced by convolution operations with kernels with less parameters.

The fully connected layer is usually the last layer applied in a CNN. However, the output of the model needs to be relevant depending on the required task. The following section explains how to tune the output of the dense layer, such that it solves a classification problem.

2.2.7 *Output of CNN*

The model needs to be designed depending on the desired task. When f_θ aims at predicting a discrete value, i.e. a *class*, the problem is a classification problem. We will only consider this case. A class is a discrete category, and each sample of the input data is assumed to belong to exactly one of the categories. The classification task is to find which category belongs each sample. To do so, we index the finite number n of classes by an integer value, from 1 to n for an n-classes classification problem. The label y_i of the input data $\mathbf{x}_i$ is therefore an integer between 1 and n.

We must design f_θ such as it outputs the class of the input sample.

The usual solution is to organize the last (dense) layer so that its output is a vector with n elements, and then to apply an $\arg\max$ function to this vector. If $\mathbf{z}$ is the output of the dense layer,

$$\hat{y} = \arg\max \mathbf{z} = \arg\max_i z_i \tag{2.2.7}$$

The drawback is that this function is not differentiable. To solve this problem, a differentiable approximation is made. It is called softmax, it is written σ and is defined as:

$$\sigma(\mathbf{z}) = \left(\frac{e^{z_1}}{\sum_j e^{z_j}}, \dots, \frac{e^{z_n}}{\sum_j e^{z_j}} \right). \tag{2.2.8}$$

The softmax function takes as input a vector of n real numbers and normalizes it into a probability distribution consisting of n probabilities proportional to the exponentials of the input numbers.

This function has the advantage of keeping the same value of the $\arg\max$ function, i.e. $\arg\max \mathbf{z} = \arg\max \sigma(\mathbf{z})$.

On Figure 2.12 is compared the two elements of the output of the softmax function with the one-hot representation of the $\arg\max$ func-

It is essential to have a differentiable approximation because for optimization the *loss* function must be differentiable. The role of the loss function is explained in the next section.

tion, ie one-hot $\circ \arg\max(\mathbf{z}) = (0, \dots, 0, 1, 0, \dots, 0)$ where the output coordinate $y_i = 1$ if and only if i is the arg max of $(z_1, \dots, z_n)$. The example is for $n = 2$.

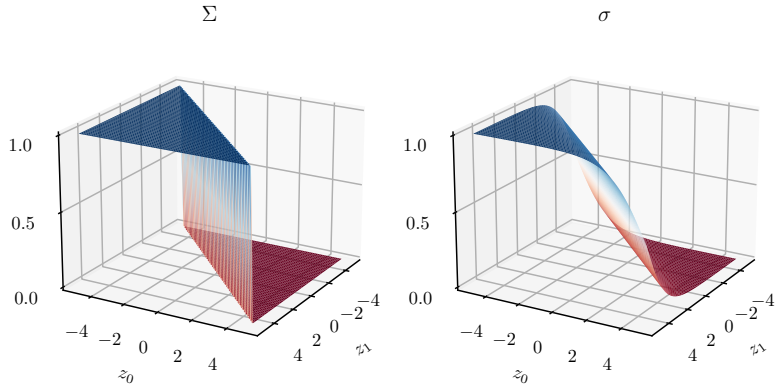


Figure 2.12: Comparaison of $\operatorname{argmax}_{i\in\{0,1\}} z_i$ (left) with the first element of $\operatorname{softmax}(z_0, z_1)$, i.e. $\sigma = \frac{e^{z_0}}{e^{z_0}+e^{z_1}}$ (right), for every pair of values of $(z_0, z_1) \in [-4, 4]^2$.

2.2.8 *Usual graph of a CNN*

Finally, we can combine all previously introduced operations to build an elementary model for a CNN composed of four layers.

First, let's define the typical layer of our model. It is a convolution operation followed by an activation function a and a pooling layer $\text{pool}(\cdot)$. The layer whose kernel of convolution is $\mathbf{k}$ can be written as $f_{\mathbf{k}} : \mathbf{x} \mapsto \text{pool}(a(\mathbf{x} \star \mathbf{k}))$.

For final classification, the models ends by a fully connected layer followed by an activation function and a softmax function σ. We note $f_{\text{fc}} : \mathbf{x} \mapsto a(\mathbf{W}\mathbf{x} + \mathbf{b})$. Finally, the entire model is written as

$$\tilde{\mathbf{y}} = f_\theta(\mathbf{x}) = \sigma \circ f_{\text{fc}} \circ f_{\mathbf{k}_3} \circ f_{\mathbf{k}_2} \circ f_{\mathbf{k}_1}(\mathbf{x})$$

where θ is the set of all optimizable variables i.e. containing $\mathbf{k}_i, \mathbf{W}$ and $\mathbf{b}$. $\tilde{\mathbf{y}}$ is described as the *predicted probabilities* of the class of the input given by the model. The *predicted class* is given by $\arg\max \tilde{\mathbf{y}}$. In order to compare $\tilde{\mathbf{y}}$ to the true label of input data $\mathbf{x}$, we need to apply a loss function.

2.2.9 *Loss function*

The *loss function*, or also called *objective function*, *criterion*, *cost function* or *error function*, is the function which translates mathematically the ability of the model to produce outputs equal to the true labels y. It penalizes situations where the output vector of probabilities diverges from the one-hot representation of the true labels, and favors the opposite case. It is again a subjective choice. For a binary classification problem, true label of sample i is written $y_i \in \{0, 1\}$ and the predicted probabilities of class by f is $p_i^0 \in [0, 1]$ (the predicted probability of class 1 is equal to $1 - p_i^0$). The more popular function is the *cross entropy*, defined by:

$$L(\{y_i\}_i, \{p_i^0\}_i) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log p_i^0 + (1-y_i)\log\left(1-p_i^0\right)\right]. \quad (2.2.9)$$

The sum is made over all the samples. An illustration of the cross entropy function is shown in Figure 2.13 for $N = 1$. We can observe that the loss function is differentiable w.r.t p and reaches its unique minimum for $p = y$.

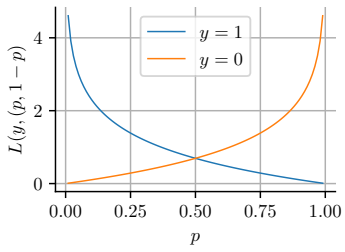


Figure 2.13: Cross entropy L between y the true label of some input $\mathbf{x}$ and $(p, 1-p)$ the output vector of probabilities given by a model for the same input $\mathbf{x}$.

This function is differentiable w.r.t. vector variable $\tilde{\mathbf{y}}$. The whole design of the graph and the loss function are made such as the loss function is differentiable w.r.t. models variables, ie θ. It makes those variables optimizable, as the following section shows.

2.3 Optimization

In general, an optimization problem consists of maximizing or minimizing a real function f by choosing input values x from within a feasible set $\mathcal{X}$.

This problem is written

$$\text{minimize } f \quad \text{subject to } x \in \mathcal{X}, \quad (2.3.1)$$

or it can be written also as $\text{minimize}_{x\in\mathcal{X}} f$. The minimum value of f is written $\min_{x\in\mathcal{X}} f$ and the input for which f is minimized is written $\arg\min_{x\in\mathcal{X}} f = x^*$.

The $*$ symbol is often added in upperscript to describe an optimal value. The f function is called the *objective* or *cost* function.

The optimization problem can be constrained or unconstrained depending on $\mathcal{X}$. The problem is unconstrained if and only if $\mathcal{X}$ is equal to the definition domain of f.

However, for future sections, we will only focus on unconstrained optimization problems, which can also be called *minimization problem*.

Solving a minimization problem can be tricky. Minimizing a function might be achieved by computing the zeros of its derivative, but it might not always be easy, especially when f is a function defined in a high dimension space. When we cannot compute the minimum analytically, the most famous solution is applying an iterative algorithm, such as the gradient descent algorithm.

2.3.1 *The concept of gradient descent*

The derivative of a function f is helpful in minimizing it because it provides the information about how to change x in order to make a slight improvement in $y = f(x)$. For example, if $f'(x) < 0$, we know that making a small positive change to x will decrease y. If we generalize for the other case, we are updating x by adding a small value that has the opposite sign of the derivative. The step needs to be relatively small as the information the derivative gives is local.

The definition of the derivative can be generalized for a function of multiple variables, and the derivative is then called gradient. The

gradient is a vector composed of all the partial derivatives of a function, where for $f : \mathbf{x} \in \mathbb{R}^n \mapsto f(\mathbf{x}) \in \mathbb{R}$, the i-th partial derivative denoted $\frac{\delta f}{\delta x_i}$ measures how f changes as only the variable x_i increases at point $\mathbf{x}$. The gradient $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ of f is defined at the point $\mathbf{x} = (x_1, \ldots, x_n)$ in a n-dimensional space as the vector:

$$\nabla f(\mathbf{x}) = \nabla_{\mathbf{x}} y = \begin{pmatrix} \frac{\partial y}{\partial x_1} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{pmatrix}. \tag{2.3.2}$$

An arrow can illustrate a gradient. It gives the direction of variation of a function. For minimizing the function f, we can again use the information given by the gradient as the negative gradient points at the steepest slope. The Gradient Descent algorithm, also called the steepest descent algorithm, proposes to update iteratively the value of $\mathbf{x}$ as $\mathbf{x}' := \mathbf{x} - \epsilon \nabla_{\mathbf{x}} y$. An example of gradient descent applied to f which is a function of 2 variables is shown on Figure 2.14.

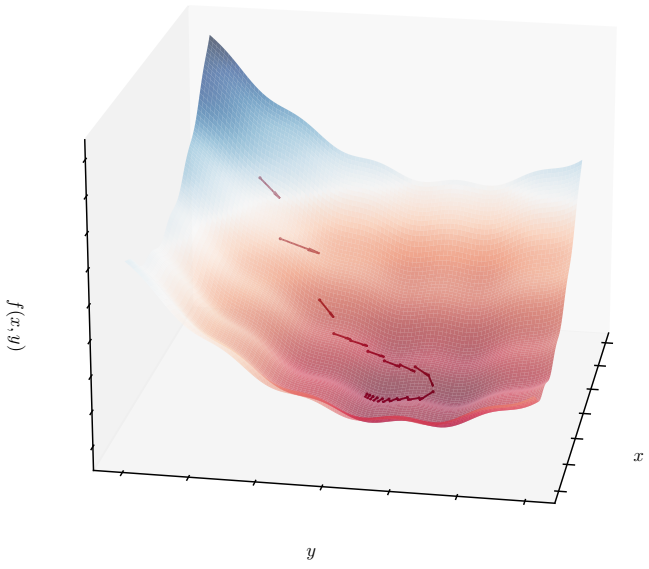


Figure 2.14: Gradient descent algorithm applied to $f(x, y) = 0.1\,(\sin 10x + \cos 10x) + x^2 + y^2$, from starting point $(-1.5, -1.5)$, with a step $\alpha = 0.1$. At each point (x_i, y_i) of the iterative algorithm, an arrow representing the gradient of f is plotted.

This fundamental algorithm is at the core of various and more complex optimization algorithms.

2.3.2 *Optimization algorithm*

The *Stochastic Gradient Descent* (SGD) is the most famous algorithm for optimization in machine learning. When the input is large (because there are many samples and/or because it is represented with many dimensions), the gradient $\nabla_\theta L(\mathbf{y}, \hat{\mathbf{y}})$ is often too costly to compute. SGD replaces the actual gradient (calculated from the entire data set) with an estimate thereof (calculated from a randomly selected subset of the

data). Especially in a big data context, it reduces the computational burden, achieving faster iterations in trade for a lower convergence rate as explained in [37].

It exists many more optimization algorithms, which avoids some critical situations encountered by SGD. However, we focus on how to compute the gradient through the CNN structure.

2.3.3 *Backpropagation in a CNN*

Let f_θ be a model parametrized by θ, and $L(\mathbf{y}, \tilde{\mathbf{y}})$ a loss function measuring the quality of the model. It compares $\tilde{\mathbf{y}} = f_\theta(\mathbf{x})$ the predicted output probabilities and $\mathbf{y}$ the true output of input train data $\mathbf{x}$. The optimization problem is $\arg\min_\theta L(\mathbf{y}, f_\theta(\mathbf{x}))$.

As explained in section 2.3.1, it can be achieved iteratively by appyling an optimization algorithm, by computing $\nabla_\theta L(\mathbf{y}, \hat{\mathbf{y}})$ in order to update θ, where θ is a vector of all optimizable parameters in the CNN.

Backpropagation, also called *backprop* is the method to compute the gradient of the loss w.r.t. θ. Backpropagation is an algorithm that applies the *chain rule* with a specific order of operations (depending on the graph) that is highly efficient.

Let x be a real number, and let f and g both be functions mapping a real number to a real number. Suppose that $y = f(x)$ and $z = g(f(x)) = g(y)$. Then the chain rule of calculus states that:

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}. \tag{2.3.3}$$

We can generalize this beyond the scalar case. Suppose that $\mathbf{x} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^n$, g maps from $\mathbb{R}^m$ to $\mathbb{R}^n$ and f maps from $\mathbb{R}^n$ to $\mathbb{R}$, then:

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}. \tag{2.3.4}$$

In vector notation, this may be equivalently written as

$$\nabla_{\mathbf{x}} z = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)^\top \cdot \nabla_{\mathbf{y}} z, \tag{2.3.5}$$

where $\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)$ is the Jacobian matrix of $\mathbf{y}$ w.r.t $\mathbf{x}$. It is the matrix of all the first-order partial derivatives of the elements of $\mathbf{y}$.

If $f : \mathbf{x} \in \mathbb{R}^n \mapsto f(\mathbf{x}) = (f_1, \dots, f_m) = \mathbf{y} \in \mathbb{R}^m$, the Jacobian matrix $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ of f at point $\mathbf{x}$ has a size of $m \times n$ and $(\frac{\partial \mathbf{y}}{\partial \mathbf{x}})_{i,j} = \frac{\partial f_i}{\partial x_j}$.

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{pmatrix} \nabla^\top f_1 \\ \vdots \\ \nabla^\top f_m \end{pmatrix} \tag{2.3.6}$$

$$= \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} \tag{2.3.7}$$

Using the chain rule, it is straightforward to express the gradient of a scalar with respect to any node in the computational graph that produced that scalar.

For example, let's take a simple layer $l = L(\mathbf{y}, \tilde{\mathbf{y}})$ where $\tilde{\mathbf{y}} = \sigma(\mathbf{x} \star \mathbf{k})$. $\mathbf{x}$ is considered as input data, $\mathbf{k}$ as an optimizable convolutional filter and σ the softmax function. $\mathbf{y}$ is the true output associated with $\mathbf{x}$ and $\tilde{\mathbf{y}}$ is the predicted probabilities. $\mathbf{x} \star \mathbf{k}$ is denoted $\mathbf{u}^{(1)}$. In order to optimize the filter, one needs to compute $\nabla_{\mathbf{k}} l$. The chain rule states that $\nabla_{\mathbf{k}} l = \left(\frac{\partial \mathbf{u}^{(1)}}{\partial \mathbf{k}}\right)^\top \left(\frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{u}^{(1)}}\right)^\top \cdot \nabla_{\hat{\mathbf{y}}} l$. The graph of this operation as well as the chain rule is given in Figure 2.15.

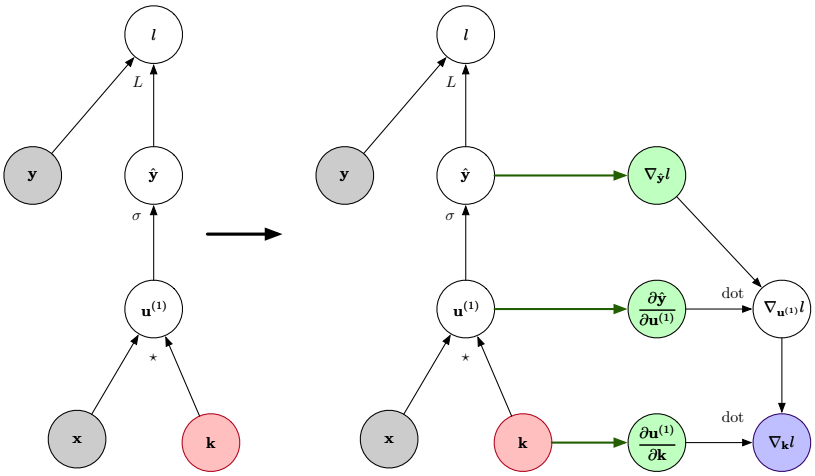


Figure 2.15: (Left) Computational graph of the operation $l = L(\mathbf{y}, \sigma(\mathbf{x} \star \mathbf{k}))$ with intermediary values $\mathbf{u}^{(1)} = \mathbf{x} \star \mathbf{k}$ and $\hat{\mathbf{y}} = \sigma(\mathbf{u}^{(1)})$. (Right) Same graph where the gradient of three nodes are computed w.r.t. their forward node. Those gradients are plotted in green. Then, combinations of those three gradients leads to the computation of $\nabla_{\mathbf{k}} l$ (blue node) needed to update the red node. Optimizable variables are in red and input data is in grey.

Now that the gradient $\nabla_{\mathbf{k}} l$ can be computed with backpropagation, we can use it to apply gradient descent to $\mathbf{k}$, with step size $\alpha \in \mathbb{R}^+$:

$$\mathbf{k} := \mathbf{k} - \alpha \nabla_{\mathbf{k}} l.$$

Now that the basis of CNN and how to optimize it were presented, here comes the practical techniques for CNN applied to steganalysis.

2.4 Deep learning for steganalysis

Convolution neural networks were firstly applied to objection recognition and image classification tasks. However, deep learning is very well suited for steganalysis, as the task is challenging. Top steganography techniques hide data very efficiently, and it requires much time to compute the good features relevant for discriminating cover and stego images. Deep learning achieves good results, but it also takes heuristics and time because it requires the design of the architecture of the CNN. The following subsections present the general hyperparameters for a model's training, then the different architectures used in chronological order.

2.4.1 *General hyperparameters*

Because it is impossible to optimize the loss over all images in the train set at the same time, we optimize it through *mini-batches*, which contains a subset of the train set. When the whole train set, split into mini-batches, has been fed to the network, we say that an *epoch* has elapsed. Often, we try to use the largest mini-batches as possible, as it is assumed to improve the convergence process.

The use of *pair training* is a hyperparameter specific to steganalysis. It consists of building mini-batches containing *pairs* of cover and stego versions of an image. Its efficiency has been recently subject to controversies, as it may help for the beginning of the training but gives

the worse performance at the end, as shown in [38].

Transfer learning is a technique to initialize the weights of a network by a pre-trained model on another task. *Curriculum learning* is a type of transfer learning to enable a faster convergence for low payloads. For an embedding rate below 0.2 bpnzAC (for JPEG images), the network might have trouble converging, as the stego is slightly changed from the cover. Curriculum learning helps the network by using higher payloads first and then decreasing the payload until reaching the desired payload.

bpnzAC stands for "bit per non-zero AC coefficients". It is the quantity often used to measure the ratio of message hidden in a JPEG image. Because there might be a lot of zeros in an image, we hide only in non-zero coefficients which are non-equal to the DC coefficient (which is the coefficient at top left for every DCT block, containing the block average value.)

2.4.2 *GNCNN*

The GNCNN, as Gaussian-Neuron CNN, is proposed by Qian et al. [39] in 2015 and is the first CNN designed for steganalysis.

It has a general architecture that inspired later designs. It starts with an image processing layer to encode the prior knowledge on the design of the network. In this layer, input data is filtered with a predefined high-pass filter K_{kv}, which is constant during training.

Then, multiple convolution layers are applied. A convolutional layer comprises three kinds of operations: convolution, non-linearity, and pooling. The non-linearity operation f is a Gaussian function, given by $f(x) = e^{-\frac{x^2}{\sigma^2}}$.

Finally, the classification module consists of three fully connected layers. The two first contains 128 neurons, followed by a Gaussian activation function, whereas the last one contains two neurons and is followed by a softmax function.

The performance of this network is close to SRM on BOSSbase [40].

$$K_{kv} = \frac{1}{12}\begin{pmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{pmatrix} \tag{2.4.1}$$

BOSSbase is a dataset of 10000 images created for the BOSS competition. It is used in all of our experiments made is this thesis. The images were obtained from never-compressed cover images coming from 7 different cameras. All images were created from full-resolution color images in RAW format (CR2 or DNG). The images were first resized so that the smaller side was 512 pixels long, then they were cropped to 512×512 pixels, and finally converted to grayscale.

2.4.3 *Xu-Net*

Xu-Net [41] is another network proposed by Xu et al. in 2017 and stands out by the use of Batch Normalization [42] (BN) layers and shortcut connections. BN is supposed to reduce internal covariate shift [42] in order to speed up the training.

It takes in input uncompressed JPEG images without the last rounding step, i.e. spatial images with floating pixels. It starts with a preprocessing layer with a DCT filter bank of size 4×4, followed by an absolute and truncation step.

Then there are 20 convolutional layers followed and a global average pooling layer. This part is responsible for learning optimized functions to transform each pre-processed input into a 384-D feature vector for classification. Batch-Normalization and ReLU follow all the convolutional layers. Several shortcut connections are applied, which are element-wise additions between two hidden layers along different depths. It makes the depth following the shortest path equals 5, whereas the longest path has 20 layers. Shortcuts favour gradient propagation by avoiding the case of *gradient vanishing*, causing inefficient training.

The problem of gradient vanishing is that in some cases, the gradient will be vanishingly insignificant, effectively preventing the weights from changing their value. In the worst case, this may ultimately stop the neural network from further training. The chain rule amplifies this phenomenon because of its multiplicative nature, so deeper networks are more prone to this issue.

The convolution kernels have a unified size of 3×3 along spatial dimensions. Pooling is achieved by convolutional layers with stride 2, after which the spatial sizes of data are cut by half and the number of channels doubles.

Its performance slightly surpasses SRM on BOSSbase.

2.4.4 *SRNet*

The philosophy of SRNet [43] is to not use any fixed pre-processing layer at the beginning of the network because fixed or constrained pre-processing kernels or kernels initialized to SRM filters or DCT bases can be detrimental for the overall network performance depending on the characteristics of the stego signal.

The overall design consists of four different types of layers, two of which involve shortcuts. The network consists of three serially connected segments: the front segment whose role is to learn effective *noise residuals*, the middle segment that compactifies the feature maps, and the last segment is a simple linear classifier. All convolutional layers employ 3×3 kernels, and all non-linear activation functions are ReLU.

The error rate of SRNet on JPEG images of size 256×256 with J-Uniward embedding at 0.4 bpnzAC is 6.70%.

2.4.5 *Efficient-Net*

In the Alaska II [44] JPEG steganalysis Challenge, many participants used the popular EfficientNet [45] pre-trained on ImageNet [46] and refined for steganalysis in the JPEG domain. It showed that CNNs trained on computer vision tasks are a good starting point for transfer learning in steganalysis. Such architectures achieved better performance than the popular SRNet.

The recent Paper [47] suggests many *surgical modifications* in order to further improve their performance for steganalysis, like removing the stride in the first layer or adding convolution blocks. They reach a wAUC of 0.9746 for 256×256 JPEG images with J-Uniward at 0.4bpnzAC on BOSSbase+BOWS2 [48] database.

This enumeration of architectures gives a brief overview of the excellent performances of deep learning to detect embedding into cover images. However, in the case of a malicious user, we will show in the next section that the security of neural networks might be at stake.

The weighted area under the receiver operating characteristic (ROC) curve (wAUC) is defined by

$$\text{wAUC} = \int_0^1 w\left(P_{\text{D}}\left(P_{\text{FA}}\right)\right) P_{\text{D}}\left(P_{\text{FA}}\right) \mathrm{d}P_{\text{FA}}$$

where $P_{\text{D}}\left(P_{\text{FA}}\right)$ is the probability of detection of a stego image as a function of the probability of false alarm, which defines the ROC curve. The weighting function $w(P_{\text{D}}) \propto 2$ if $P_{\text{D}} < 0.4$ and $w(P_{\text{D}}) \propto 1$ if $P_{\text{D}} \geq 0.4$ normalizes the wAUC to be in the interval $[0, 1]$.

2.5 A weakness of CNN: adversarial examples

We saw how differentiability is an essential notion in deep learning because it allows the optimization of the parameters of the networks by backpropagation. Nevertheless, it has a notable drawback: this makes the classifiers easily subject to attacks.

Szegedy et al. [49] made an intriguing discovery in 2013: several machine learning models are vulnerable to *adversarial examples*. These machine learning models misclassify only slightly different examples

from correctly classified examples drawn from the data distribution by computing an adversarial noise by differentiation. It makes those classifiers very vulnerable to malicious users. The usual illustration of this phenomenon is given in Figure 2.16. The Figure shows a hyperparameter of adversarial perturbation, which can be small (left) or large (right), which makes the example closer or further from the decision boundary.

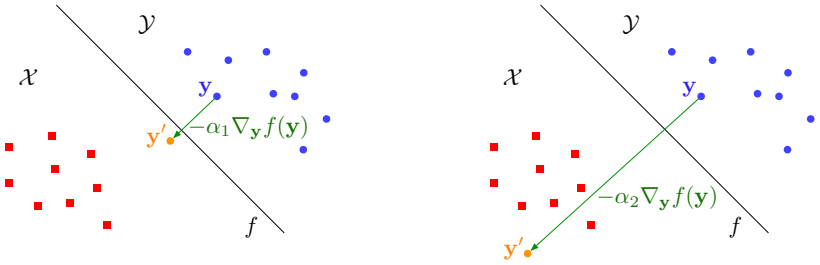


Figure 2.16: Usual illustration of an adversarial example, where a classifier f, discriminating between red $\mathcal{X}$ and blue $\mathcal{Y}$ distributions, is fooled by the orange point $\mathbf{y}'$ obtained from $\alpha_1 \nabla_{\mathbf{y}} f(\mathbf{y})$ (left) or $\alpha_2 \nabla_{\mathbf{y}} f(\mathbf{y})$ (right) subtracted to initial point $\mathbf{y}$. $\mathbf{y}'$ is classified as belonging to $\mathcal{X}$ whereas it should be classified in the other class. Here $\alpha_1 < \alpha_2$.

In steganalysis, this is a great opportunity for Alice as she wants to avoid Eve's detectors. For example, in Figure 2.17 are taken a cover image $\mathbf{x}$ and a stego $\mathbf{y}$ obtained from it which are correctly classified by f a detector. Alice could compute the adversarial noise $\nabla_{\mathbf{y}} f(\mathbf{y})$ and subtract it (after multiplying by a factor α) to the stego such as it will be misclassified.

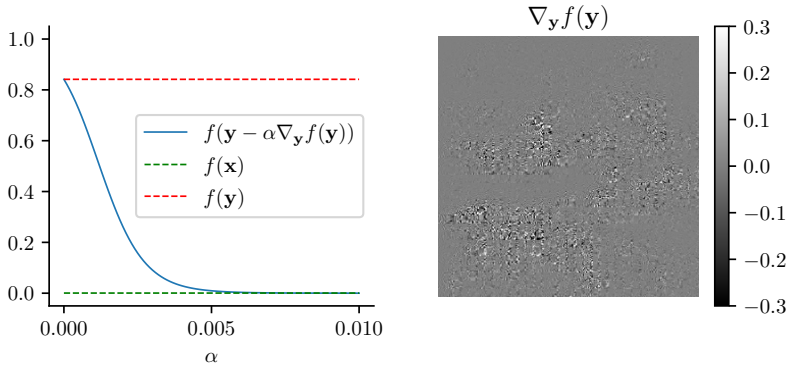


Figure 2.17: For f a classifier which outputs the stego class probability, (left) evaluation of adversarial images $\mathbf{y} - \alpha \nabla_{\mathbf{y}} f(\mathbf{y})$ by f w.r.t size of step α in x axis, compared to evaluation of a cover $\mathbf{x}$ classified in cover class and a stego $\mathbf{y}$ classified in the stego class by f (dashed lines). (Right) Plot of the gradient $\nabla_{\mathbf{y}} f(\mathbf{y})$.

It gives a way to produce adversarial vectors but which are neither *images* (because the noise contains continuous values which are not quantized or rounded) neither *stegos* (because adding a noise will destroy the embedding of the message, which is the initial purpose of steganography). In 2019 there is a paper that proposes an alternative named ADV-EMB to produce *adversarial stego images*, evoked in section 1.5.10. It is an algorithm that we will use in our experiments.

2.6 Conclusion of the chapter

We studied in this chapter the elementary bricks of Deep Learning and a particular type of model, Convolutional Neural Networks, adapted to computer vision tasks. The operation of 2D convolution decreases the

number of parameters to learn. It is well suited to the grid structure of an image too.

We also learned the fundamental algorithm of Gradient Descent which is at the heart of optimization processes to learn the parameters of a network. Their application to deep learning models is achieved by backpropagation. It is possible because the networks are differentiable functions.

We finally presented in a historically order four structures designed to perform steganalysis. Over the years, the architectures for steganalysis tasks became better performing. At the date of the manuscript, the best models are pre-trained on large databases then finetuned to steganalysis tasks.

However, deep learning models are not robust against malicious attacks. It represents the perfect opportunity for the steganographer to take advantage of CNN's drawbacks. ADV-EMB is an algorithm adapted to steganography that adapts the distortion function to avoid a differentiable classifier. In the next chapter, we see how sequentially Alice can construct a better embedding function undetectable by a whole range of classifiers.

The iterative minmax protocol

3

Traditionally, each new publication shows better performance measured by the latest techniques just created by the opponent. For example, HUGO [21] aims at avoiding the SPAM [22] detector, and a few years later, Rich Models [32] tries to detect HUGO. As shown in Figures 1.20 and 1.21, Alice and Eve have been iterating between each other to build better and better embedding functions and detectors for years of research. The whole idea of our first contribution is to build an algorithm that will play this iterative scheme automatically.

We saw in previous chapters that it exists:

- steganalysis based on CNN to detect stego images. Machine learning enables the detection of any embedding function at the cost of the learning process through examples and good architecture design.
- steganographic schemes which rely on fooling directly a specific detector f, like ASO does by adjusting its distortion w.r.t. the output of f. This technique, just like ADV-EMB [50], is called an *adversarial embedding scheme*.

Using both of those methods may seem like an endless game, where Alice and Eve could infinitely adapt their method depending the one chosen by their opponent, as shows Figure 3.1. It is precisely the context of *game theory*, where two opponents with an antagonist objective play a game.

We propose an algorithm inspired by the methodology of game theory. It allows Alice to simulate the game played between her and Eve. It is general and can be applied to any pair of adversarial embedding schemes / differentiable detectors. The experiments of this chapter use the adversarial attack presented in section 3.1 as a potential candidate, and two different CNNs, namely XU-Net [41] and SRNet [43], as detectors.

Under the condition that the set of detectors that Alice assumes Eve to have is sufficiently rich (e.g. CNNs), and that she has an algorithm enabling to avoid detection by a single classifier (e.g. adversarial embedding [50], Dynamic programming based Syndrome Trellis Code [51]), we show that the proposed algorithm converges to an efficient steganographic algorithm. It is made possible by using a min max strategy which consists at each iteration in selecting the least detectable stego image for the best classifier among the set of Eve's learned classifiers.

The structure of this chapter is as follows:

- We firstly give more details about the ADV-EMB method.
- We introduce key notions of game theory useful to build our proposed algorithm.

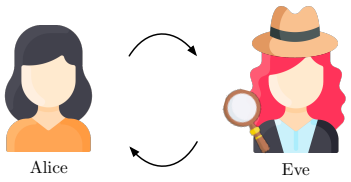


Figure 3.1: Endless game between Alice and Eve who can elaborate new methods.

- We analyze theoretical and practical convergence issues with links to other iterative strategies (see section 3.3.1);
- We propose a way to attack a set of classifiers with possibly different architectures. The set is combined using an appropriate new calibration function, which further improves both the quality and versatility of the algorithm;
- We address the problem of Curriculum Learning and show how bad learning can jeopardise the performances of the steganographic scheme.
- We also propose an extended evaluation of the protocol using different quality factors, embedding rates, initial distortion functions,
- We consider embedding in the spatial domain, using HILL as an embedding scheme, and we show that the behaviour of the proposed algorithm is consistent with what is obtained in the JPEG domain.

3.1 The ADV-EMB scheme

The pioneering work of Szegedy et al. [49] demonstrated that classifiers based on neural networks could be forced to misclassify an image by adding a specific signal of small amplitude, as it was introduced in section 2.5. While in the field of computer vision, the attacker has the freedom to change the image, attacking a steganographic detector is more difficult due to the constraint that the resulting stego image has to carry a particular message. Note that this property was known in steganography long before (see [19], [24] and recently also [51]).

A method ADV-EMB inspired by those in the field of adversarial learning was proposed in [50], and due to low computational complexity, it is used in this work. ADV-EMB modifies costs of DCT coefficients such that changes of coefficients during embedding are correlated with the gradient of the soft output of a CNN steganalyzer, making the image less detectable. Since the embedding function h_{emb} is not differentiable with respect to costs, ADV-EMB comes with an heuristic which consists in modifying costs ρ_i^+ and ρ_i^- of some existing cost function ρ (e.g. J-Uniward, UERD, see section 1.5) in the following way:

$$\rho_i^{+,new} = \begin{cases} \rho_i^+/\alpha & \text{if } \frac{\partial f}{\partial x_i}(\mathbf{x}) < 0, \\ \rho_i^+ & \text{if } \frac{\partial f}{\partial x_i}(\mathbf{x}) = 0, \\ \rho_i^+\alpha & \text{if } \frac{\partial f}{\partial x_i}(\mathbf{x}) > 0, \end{cases} \tag{3.1.1}$$

and

$$\rho_i^{-,new} = \begin{cases} \rho_i^-/\alpha & \text{if } \frac{\partial f}{\partial x_i}(\mathbf{x}) > 0, \\ \rho_i^- & \text{if } \frac{\partial f}{\partial x_i}(\mathbf{x}) = 0, \\ \rho_i^-\alpha & \text{if } \frac{\partial f}{\partial x_i}(\mathbf{x}) < 0, \end{cases} \tag{3.1.2}$$

where $\frac{\partial f}{\partial x_i}$ is the partial derivative of f with respect to the value of the i^{th}-DCT coefficient at its current value x_i and α is a parameter set to recommended value 2.

The partial derivative of f is computed for a semi-stego object $\mathbf{s}$ (in which a portion of the message is embeded) such as the final stego

object $\mathbf{y}$ is assigned to a low probability of being stego (i.e. a small $f(\mathbf{y})$) after embedding. Therefore, for instance if $\frac{\partial f}{\partial s_i}(\mathbf{s}) > 0$, a positive increment on s_i would increase this probability and consequently, this situation is penalized by increasing the corresponding modification cost ρ_i^+ by a factor α.

Since steganalyzers are usually not good models of cover images,[1] modulating costs of all coefficients would probably lead to very detectable models. The solution adopted by ADV-EMB is to dispatch DCT coefficients into *common* and *adjustable* groups, $\mathcal{L}_c$ / $\mathcal{L}_a$, corresponding to $(1-\beta)$ / β fractions of coefficients, and then modify only coefficients in the adjustable group. The image coefficients belonging to each group at chosen at random. By minimizing β, ADV-EMB changes costs of a minimal number of coefficients. ADV-EMB finds the minimal β by exhaustive search in $\beta \in \{0.1, 0.2, \ldots, 1.0\}$. The gradient used to modulate costs is calculated after coefficients from the common group are used for embedding a $1-\beta$ fraction of bits of the message m. The procedure, for a given ratio β, to obtain the stego $\mathbf{y}$ from the cover $\mathbf{x}$ with an initial symmetric cost map $\{\rho_i^{\pm 1}\}_i$, is illustrated in Figure 3.2.

[1] Steganalyzer models discriminate cover from stego images, but they do not model cover images themselves.

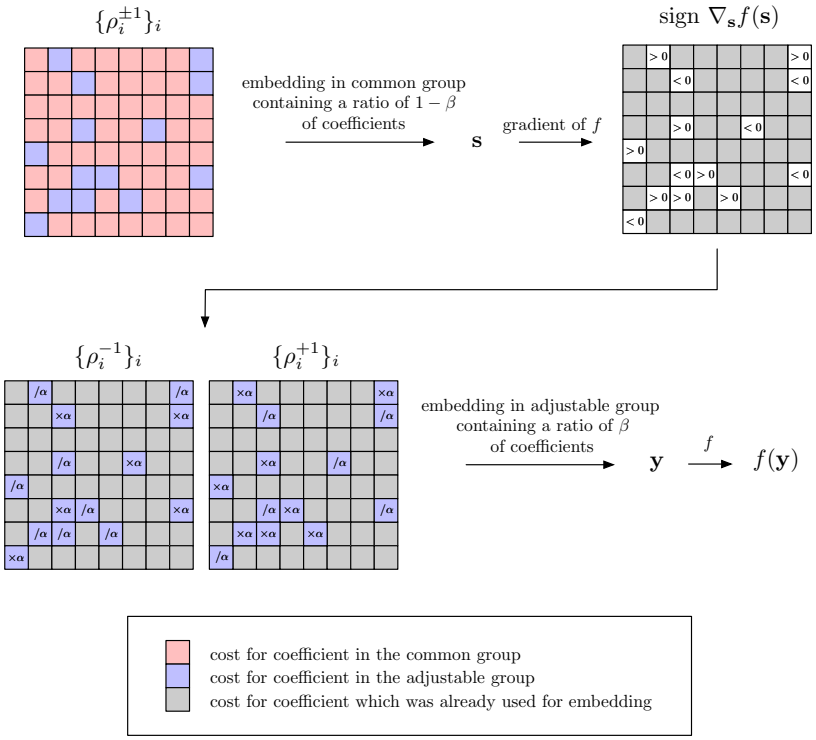


Figure 3.2: Scheme of ADV-EMB, for a given ratio β. At the beginning the cost map $\{\rho_i^{\pm 1}\}_i$ is symmetric. All costs are dispatched in two different groups: common or adjustable. Firstly a ratio of $1-\beta$ of the message in embedded in the common group to give an image $\mathbf{s}$. Then the gradient of the output of $\mathbf{s}$ given by f w.r.t. coefficients s_i is computed. The sign of the gradient is used to udpate the value of the costs in the adjustable group given Equations 3.1.2,3.1.1, such as the costs map $\{\rho_i^{+1}\}_i$, $\{\rho_i^{-1}\}_i$ are no longer symmetric. Finally the rest of the message is embedded in the adjustable group. Finally we obtain the stego image $\mathbf{y}$ which contains all bits of the message.

This heuristic helps decrease the output of a classifier. In Figure 3.3 is given an example of the adversarial impact of ADV-EMB to defeat a trained classifier f, w.r.t. the ratio β of coefficients in the adjustable group. For a ratio $\beta = 0$, all image coefficients are in the common group (the cost map is the initial non-modified one); stegos are detected on average by f (as expected). However, for $\beta > 0$, the detectability is decreased. It can be seen as an adversarial impact on f,

as we saw in Figure 2.17. Here we finally obtain an adversarial object which is a *stego image*.

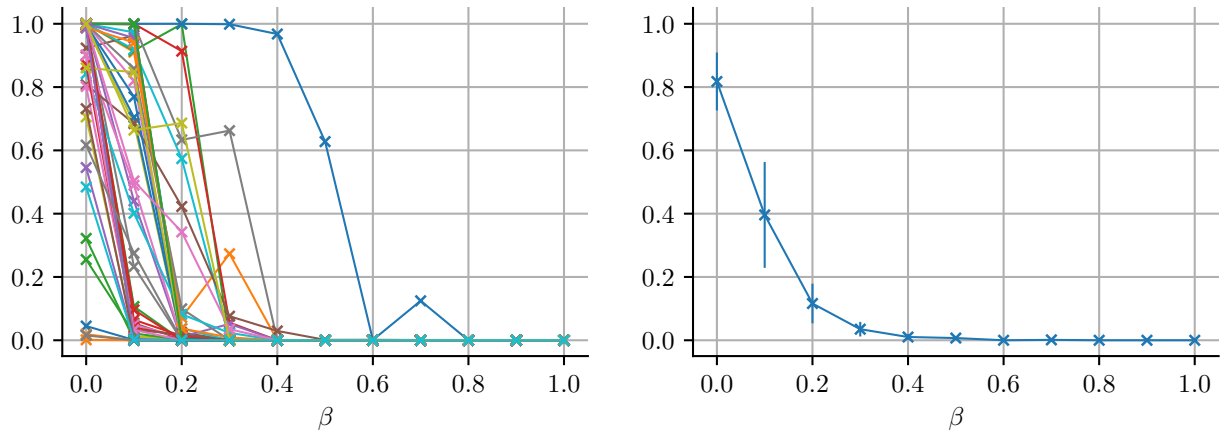


Figure 3.3: (Left) Detectability of 50 stegos images produced with ADV-EMB with $\beta \in \{0, 0.1, \ldots, 0.9, 1\}$ against a classifier f trained to detect J-Uniward on 512×512 images at QF 75 with a payload of 0.4 bpnzAC and (right) average and variance of the same detectability over 100 images.

Although a significant question here stands. *What is next ?* We can imagine that a newly trained classifier could easily detect the attack produced by ADV-EMB. That is the whole point of our contribution, which aims at iterating by producing stegos secured w.r.t. a class of classifiers. To do so, we must first introduce basic notions of game theory.

3.2 The steganographic game

3.2.1 *Reminders on game theory*

Game theory is the mathematical study of interaction among independent, self-interested agents, also called *players*. It has been applied to many disciplines to model competing agents' behaviours, like in economics or biology. It applies when each agent can choose an action among a set of possible actions, and the outcome depends on the combination of the actions taken by all agents independently. Self-interested agents mean that each player has their vision of the states of the world they like and wants those to happen.

As explained in book [52], the dominant approach to model an agent's interests is utility theory. This theoretical approach aims to quantify an agent's degree of preference across a set of available alternatives. This can be achieved by defining a utility function, which is a mapping from states of the world to real numbers. These numbers are interpreted as measures of an agent's level of satisfaction in the given states.

Games with n players (having discrete actions) can be entirely described by a $n+1$-dimensional table storing the utilities of each player (contained in the first dimension) and all possible combinations of actions of all n players described as the intersection of n rows in the last n dimensions.

The first dimension contains n rows, and every other dimension contains as many rows as the number of actions available to each player.

The difficulty of game theory comes when two agents may have

different utility functions, like in the simple two-players game presented in Figure 3.4. Here the maximum reward is not reached for the same couple of actions of both players: the first player maximizes his utility for actions (a_0, b_2) whereas the second one for (a_1, b_1).

$u_1(a_i, b_i)$

	b_0	b_1	b_2
a_0	-1	2	4
a_1	1	-3	-2

$u_2(a_i, b_i)$

	b_0	b_1	b_2
a_0	0	1	-3
a_1	-1	2	-2

Figure 3.4: Description of a two players game where first player 1 has two actions (a_0, a_1), second player 2 has three (b_0, b_1, b_2), and $u_1, u_2 : (a_0, a_1) \times (b_0, b_1, b_2) \to \mathbb{R}$ are the utility functions of two players.

In a two-players zero-sum game, the sum of both utilities are equals to 0, such as a gain for a player drives to a loss for the other. It is the case of the game presented in Figure 3.5.

$u_1(a_i, b_i)$

	b_0	b_1	b_2
a_0	-1	2	4
a_1	1	-3	-2

$u_2(a_i, b_i) = -u_1(a_i, b_i)$

	b_0	b_1	b_2
a_0	1	-2	-4
a_1	-1	3	2

Figure 3.5: Game with same player 1's utility than in Figure 3.4, but where this time utility of second player u_2 is equal to $-u_1$.

In the case of a two-player zero-sum game in which players' actions are discrete, the game can be described entirely with a single table where the rows and columns are for actions of both players, and cells contain the utility of one of the players.

When actions are continuous, utility functions can be displayed as a surface in a 3D space.

Game theory's view is interesting in our problem because Alice and Eve can be seen as two players with antagonistic goals (like in a zero-sum game), and the performance of each player depends on the action of the other.

3.2.2 *The steganographic game*

We define the normal-form of the steganographic game (following the model defined in [52]):

Definition 3.2.1. *the steganographic game, denoted $\mathcal{G}$ is a tuple $(\mathfrak{N}, m, \mathcal{A}_a, \mathcal{A}_e, u)$ where :*

- *$\mathfrak{N}$ is a set of 2 players, indexed by p where $p \in \{a, e\}$ (for Alice and Eve)*
- *m is the length of the message to embed in each image*
- *$\mathcal{A}_a, \mathcal{A}_e$ is a possibly infinite set of actions of Alice and Eve.*
  - *$\mathcal{A}_a$ is the set of all embedding functions $h_{\text{emb}} : \mathbf{x} \in \mathcal{X} \mapsto \mathbf{y} \in \mathcal{Y}$ which maps a cover to a stego image, obtained by the embedding of a message of size m*

  - *$\mathcal{A}_e$ is the set of all possible detectors f providing a stego probability class, which discriminates cover from stego images.*

- *$u = (u_a, u_e)$ where $u_p : \mathcal{A}_a \times \mathcal{A}_e \to \mathbb{R}$ is a real-valued utility function for player p. For a couple of action $(h_{\text{emb}}, f) \in \mathcal{A}_a \times \mathcal{A}_e$, Eve's utility is defined as the accuracy of classification under equal prior, i.e. the average of the true positive rate and true negative rate, given by*

$$u_e(h_{\text{emb}}, f) = \mathbb{E}_{\mathbf{x}\sim P_{\mathcal{X}}}[f(\mathbf{x}) < 0.5] + \mathbb{E}_{\mathbf{y}\sim \mathcal{P}_{\mathcal{Y}}^{(h_{\text{emb}})}}[f(\mathbf{y}) \geq 0.5]. \quad (3.2.1)$$

  Alice's utility is the opposite of Eve's utility because her goal is to fool Eve: $u_a = -u_e$.

Other scores such as FP50 [53] (the false positive rate when the false negative rate equals 50%) or MD5 [54] (the miss detection rate when the false positive rate equals 5%) could also be used.

This steganographic game is a zero-sum game, i.e. for all action profile $(h_{\text{emb}}, f) \in \mathcal{A}_a \times \mathcal{A}_e, u_a(h_{\text{emb}}, f) + u_e(h_{\text{emb}}, f) = 0$.

The usual vizualization of the game in the table form is shown on Figure 3.6, where the cell at the intersection of row h^i_{emb} with column f^j contains value $u_e(h^i_{\text{emb}}, f^j)$.

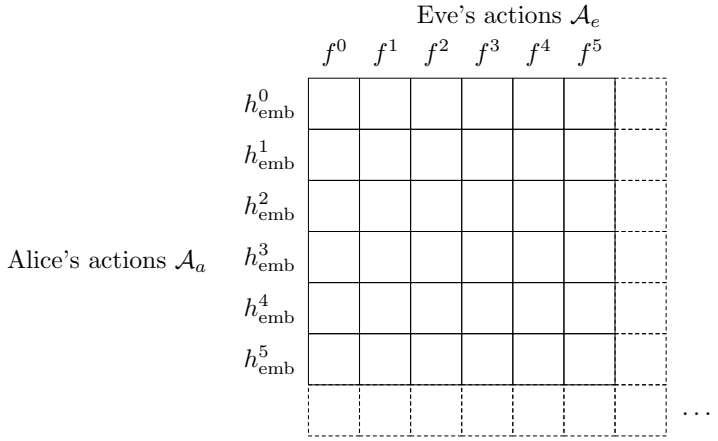


Figure 3.6: The steganographic game where the rows are Alice's actions (the embedding functions) and columns for Eve's actions (the classifiers). The dashed cells show that the table is very huge and it has been reduced for illustration.

For a given game, each player thinks of which actions to take to maximize their utility, given that the outcome might be uncertain. The planning of actions is called a *strategy*.

3.2.3 *Nash equilibrium and the minmax strategy*

Given a game $\mathcal{G}$, each player can elaborate a *strategy*. A player's strategy is the action that the player chooses to play for a given setting, where the outcome depends on the actions of all players at the same time.

We will limit ourselves in this work to *pure strategies* which consist in selecting a single action and playing it. Other kinds of existing strategies are *mixed strategies*, where a player can randomise over the set of available actions according to some probability distribution.

In 1950 is introduced by John Nash the most influential solution concept in game theory, the *Nash equilibrium*. Intuitively, a Nash equilibrium is a stable strategy: no agent would want to change his strategy if he knew what strategies the other agents were following.

Nash equilibria in zero-sum games can be viewed graphically as a *saddle* in a high-dimensional space as shown in Figure 3.7 (in a 3D space with two players). At a saddle point, any deviation of any agent lowers his utility and increases the utility of the other agent.

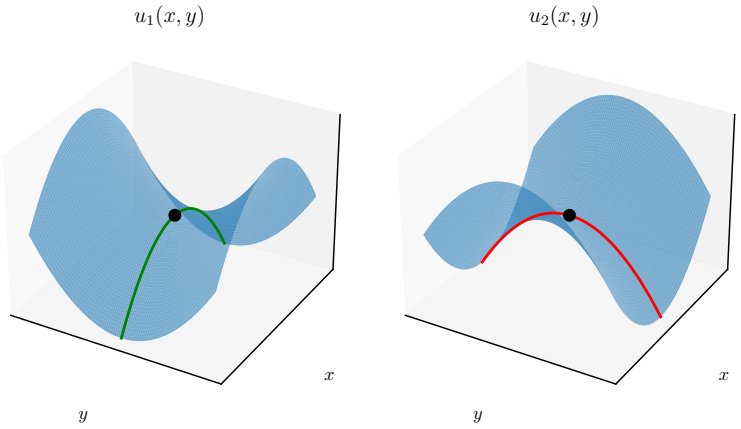


Figure 3.7: Nash equilibrium in the two players zero-sum game described by the utilities of both players u_1 and $u_2 = -u_1$ is the saddle point noted by a black dot. For this given couple of actions (x_0, y_0), the colored lines show that any change of action of each player will decrease their utility (given in the left for first player and in the right for second).

The min max strategy is another strategy that maximizes player's worst-case payoff in the situation where all the other players happen to play the strategies which cause the most significant harm to him. The theoretical min max strategy in two players' zero-sum games equals the Nash equilibrium. It is why min max was the strategy chosen in our contribution. Because we only consider pure strategies[2] in this work, there is no equality with the Nash equilibrium.

[2] Future possible works should consider mixed strategies in order to play Nash equilibrium.

Because of the Kerckhoffs' principle presented in section 1.4.1 which assumes a worst case attack from Eve, we chose in this work that Alice should consequently select an algorithm $h_{\text{emb}} \in \mathcal{A}_a$ minimizing the utility of Eve's best detector. Alice consequently wishes to solve the following optimization problem:

$$\arg\min_{h_{\text{emb}} \in \mathcal{A}_a} \max_{f_{\text{det}} \in \mathcal{A}_e} u_e(h_{\text{emb}}, f). \tag{3.2.2}$$

This optimization problem in (3.2.2) is challenging to solve because the expectation in the utility function does not have an analytical formula (because distributions are unknown), also because the inner maximum and outer minimization are both over infinite sets. To make the problem tractable, we propose an iterative protocol in order to approach this solution.

3.2.4 *Solving the simple steganographic game*

The algorithm we propose, displayed in Figure 3.8, comes for initialisation with an initial embedding function and a detector. At each further iteration, both players create a new action. Alice begins by avoiding Eve's best detector among existing actions, according to a min max strategy. Then Eve plays second by creating a new detector trying to detect the just created stegos by Alice.

We utilise the fact that when Alice is searching for a suitable algorithm, the classifier in (3.2.2) does not have to be workable in practice. Specifically, the classifiers are (unreasonably) assumed to be selected for each image separately in this work. This assumption is not realistic and overly pessimistic[3] for Alice since it upper bounds the actual

[3] It is unlikely that Eve should be able to train a perfect classifier selector that maps each image to the best classifier she possesses for this image.

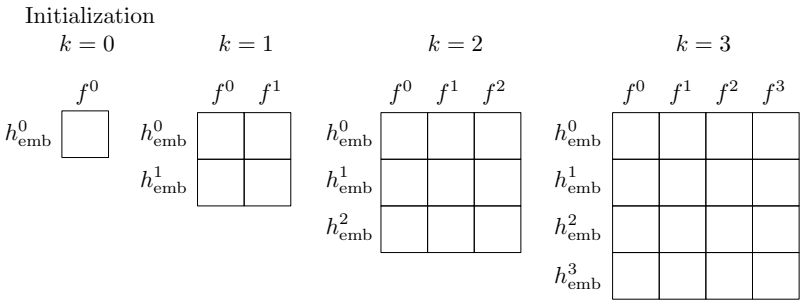


Figure 3.8: Building an iterative game where the number of actions for each player increases.

detectability achievable by Eve, but it decreases the computational complexity.

At k^{th} iteration, the proposed protocol consists on the two following macro-steps :

Data: $\mathcal{Z}^0, \cdots, \mathcal{Z}^{k-1}$, stego bases, $\mathcal{X} = \{\mathbf{x}_{(1)}, .., \mathbf{x}_{(N)}\}$ cover base, and set of detector $\mathcal{F}^{k-1} = \{f^0_{\text{det}}, \cdots, f^{k-1}_{\text{det}}\}$.

1. Creation of a stego set $\mathcal{Y}^k$ by using an embedding function h_{emb} maximally secure with respect to the set of detectors $\mathcal{F}^{k-1} = \{f^0_{\text{det}}, f^1_{\text{det}}, \ldots, f^{k-1}_{\text{det}}\}$, i.e. for a given image $\mathbf{x}$ it uses function
$$h_{\text{emb}} = \underset{h_{\text{emb}} \in \mathcal{A}_a}{\arg\min} \max_{f \in \mathcal{F}^{k-1}} f(h_{\text{emb}}(\mathbf{x})); \tag{3.2.3}$$
2. Creation of a new detector f^k_{det}, which should be optimal for stego images produced in step 1 and appends it to the pool, i.e. $\mathcal{F}^k = \mathcal{F}^{k-1} \cup \{f^k_{\text{det}}\}$.

Algorithm 2: Two macro-steps of the k^{th} iteration of the proposed protocol.

This protocol 2 is very general. It is at the core of both contributions. The algorithm 3 is the practical application of it. For example, it leverages ADV-EMB to solve Equation 3.2.3.

Notice that the distortion function/embedding algorithm is not fixed, but it is implicitly defined by the set of detectors $\mathcal{F}^k$ and a set of $\mathcal{A}_a$ of Alice's strategies. It means that for each image, Alice picks the most secure algorithm for a given image with respect to detectors $\mathcal{F}^k$ she believes Eve might own.

The above algorithm 2 is general, as Alice can use any set of embedding functions, $\mathcal{A}_a$, even those evading a specific detector as discussed in the previous section, and she can assume any set of steganographic detectors $\mathcal{A}_e$. In practice, one can guess that the larger both sets are, the more secure the resulting algorithm will be. The only caveat is that, due to the operation consisting in minimizing output of detectors in Equation (3.2.3), detectors should have comparable outputs. This problem of calibration is described in detail in section 3.5.2.

The very good property of convergence of this protocol is shown in the next section.

3.3 Theoretical properties

3.3.1 *Convergence of the algorithm*

The following theorem proves the convergence of our protocol under mild conditions on $\mathcal{F}$.

Theorem 3.3.1. *Let $\mathcal{F} = \{f : \mathcal{X} \to \mathbb{R}\}$ be a set of functions and let $\mathcal{F}^1, \mathcal{F}^2, \ldots, \mathcal{F}^k, \ldots$ be a sequence of subsets such that $\mathcal{F}^1 \subset \mathcal{F}^2 \subset \ldots \subset \mathcal{F}^k \subset \ldots \subset \mathcal{F}$. Suppose all functions $f \in \mathcal{F}$ are bounded by some constant c, i.e. $(\exists c \in \mathbb{R})(\forall f \in \mathcal{F})(\forall \mathbf{x} \in \mathcal{I})(f(\mathbf{x}) \leq c)$.*

Then the limit $\hat{f}(\mathbf{x}) = \lim_{k\to\infty} \max_{f\in\mathcal{F}^k} f(\mathbf{x})$ exists.

Proof. Define function $f^k_{\max}(\mathbf{x}) = \max_{f\in\mathcal{F}^k} f(\mathbf{x})$. Then for every $\mathbf{x} \in \mathcal{I}$ the sequence $f^1_{\max}(\mathbf{x}), f^2_{\max}(\mathbf{x}), \ldots, f^k_{\max}(\mathbf{x}), \ldots$ is non-decreasing and because of the boundedness assumption $\forall f \in \mathcal{F}$, $f(\mathbf{x}) \leq c$, the sequence is bounded by c as well. The monotone convergence theorem then states that the sequence $f^k_{\max}(\mathbf{x})$ converges to some value, which is denoted by $\hat{f}(\mathbf{x})$, which proves pointwise convergence of $f^k_{\max}$ to $\hat{f}$. □

The above theorem implies that, when k is large, the maximization w.r.t. $f \in \mathcal{F}^{k-1}$ is replaced by $\hat{f}$ (or a function ϵ-close to $\hat{f}$). The algorithm defines detectability $\hat{f}(x)$ as a limit

$$\hat{f}(\mathbf{x}) = \lim_{k\to\infty} \max_{f\in\mathcal{F}^k} f(\mathbf{x}). \tag{3.3.1}$$

Note that the security of the resulting steganographic algorithm depends on two factors: (i) the set of all possible detectors $\mathcal{F}$; (ii) the attacking quality on the classifier $f \in \mathcal{F}$. Thus, improving any of them should improve the quality of the scheme.

Theorem 3.3.1 assumes functions $f \in \mathcal{F}$ to be bounded. This condition can be trivially ensured for any function based on machine learning classifiers, as they are already bounded (e.g., Neural Networks), or they can be trivially bounded by applying some scaling or passing their output through abounded and monotonous functions like sigmoid or tanh .

Furthermore, the usual functions involved in a neural network are not only bounded but also Lipschitz continuous. Indeed, dot product,[4] convolution, max pooling, ReLU, sigmoid or tanh are all Lipschitz continuous, and the sum and composition of such functions also are, i.e., neural networks are Lipschitz continuous. This observation leads to a stronger form of convergence.

[4] To make sure that each instance of an architecture has the same (maximal) Lipschitz constant, it is sufficient to add a regularisation term to the objective function.

If each $f \in \mathcal{F}$ is Lipschitz continuous with common constant, then it is known that $\hat{f}$ is also Lipschitz continuous with the same constant provided that $\hat{f}$ achieves a finite value for some $\mathbf{x}$. Since $\hat{f}$ is bounded, it is finite everywhere and thus Lipschitz continuous.

In addition, since all functions $f^k_{\max}$ and function $\hat{f}$ are defined on a compact subset of $\mathbb{R}^{H\times W}$ and the sequence $f^k_{\max}$ is monotonically increasing then Dini's theorem [55] applies which gives uniform convergence.

Roughly speaking, uniform convergence indicates that the series of functions on which the min-step of the algorithm operates converges everywhere in the input space, at least with a rate that does not depend on $\mathbf{x}$. Although this rate lower bound is unknown, it can be argued that the algorithm could be stopped when $\|f^k_{\max} - f^{k-1}_{\max}\|$ is no greater than a given threshold. However, given the stochasticity of neural networks training and the time spend on it, it appears safer to stop after a predefined number of iterations as proposed in Algorithm 3.

3.3.2 *Connections with generative adversarial networks*

The proposed algorithm is to some extent related to a popular machine learning tool known as Generative Adversarial Networks (GANs) [56], which has been recently used in steganography to learn the cost function [57]. The differences between Algorithm 3 and GANs are discussed below.

In theory [58], the solution of GANs corresponds to a Nash equilibrium of the zero-sum game with the cost function defined in (3.2.2). The main difference between GANs [57] and our contribution is therefore in the generator.

In [57], the generator is implemented *explicitly* by an optimized network assigning costs to individual coefficients, which are then used in the embedding simulator. It is in sharp contrast to this contribution, where the generator is *implicit* and corresponds to an attack of a particular classifier (or a set of them). Therefore, this algorithm's convergence (or training) should be more straightforward, more stable, and Alice is saved from the hassle of designing an architecture for the generator. Indeed, Ref. [57] reports the error of detection by XU-Net of a GAN JPEG stego images with payload 0.4 bpnzAC equal to 11.8%, whereas that of the steganography proposed here is equal to 20.7%, which makes the proposed approach twice more secure.

However, an advantage of GAN-based steganography is that it can directly estimate embedding costs, which the proposed scheme in this section cannot do yet, but it is solved in the next chapter.

3.4 Implementation of the protocol

Because the protocol proposed in previous section 3.2.4 is very general, the following two subsections discuss particular choices of Alice's and Eve's strategies used in this contribution.

3.4.1 *Alice's strategy*

In our experiments, the set of steganographic algorithms used by Alice is the union of ADV-EMB attacks against all Eve's detectors $f_{\text{det}} \in \mathcal{A}_e$ with $\beta \in \{0.1, 0.2, 0.3, \ldots, 0.9, 1.0\}$ and J-Uniward. The stego retained by ADV-EMB is the one minimizing β and which crosses the boundary decision of cover/stego of a detector. This set of algorithms has an infinite size theoretically. But if the optimality of ADV-EMB attack

against a given detector f_{det} is assumed, it is sufficient during the k^{th} iteration to consider attacks against a limited set of detectors $\mathcal{F}^{k-1} = \{f^0, \dots, f^{k-1}\}$, as we cannot do better against this set. Thus, the min max problem in step 1 of each iteration is over a finite set, and hence computationally feasible.

An important implementation detail here is that stego images created by attacking $\mathcal{F}^{k-1} = \{f^0, \dots, f^{k-1}\}$ and outputs of all detectors on them can be cached and used in subsequent iterations. This means that at every iteration, Alice needs to (i) create (adversarial) stego images against the detector f^{k-1} appended to $\mathcal{F}^{k-1}$ in the previous iteration, and (ii) calculate outputs of f^{k-1} for stego images created in iterations $1, 2, 3, \dots, k$. This setting significantly decreases the computational complexity.

Here, we emphasize on the differences w.r.t. strategies proposed in [50] which we call *last iteration* and *random* strategies:

- In *last iteration* strategy, Alice's embedding algorithm is ADV-EMB attacking only the last trained detector f^{k-1}.

- In *random* strategy, Alice's embedding algorithm is ADV-EMB attacking a detector $f \in \mathcal{F}^k$ where each stego image of the training set is sampled uniformly over the previous iterations.

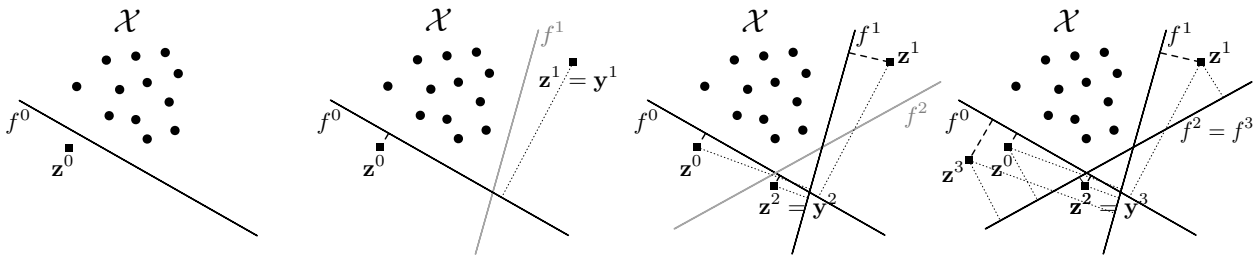
Again, note that Alice behaves more strategically (and conservatively), as she uses the algorithm producing the least detectable stego image by an unrealistic detector (in the sense that we assume that Alice knows a piece of information not accessible in practice). The rationale here is the fact that for the next iteration, Eve might learn a better detector than the last trained, for example by training a new classifier from the ensemble of already trained classifiers.

3.4.2 *Assumed Eve's detectors*

The term *Eve's detectors* is ambiguous, as it can refer to the set of detectors $\mathcal{A}_e$ used by Alice during the optimization of Equation (3.2.2), and the set of detectors used by real Eve to detect Alice's steganography. The mismatch between these two can be devastating and the whole history of steganography is precisely about this mismatch. To clarify, the former will be called *assumed Eve's detectors* and denoted $\mathcal{A}_e$ and the latter *real Eve's detectors* denoted $\tilde{\mathcal{A}}_e$.

In the optimization of Equation (3.2.2), the set of detectors $\mathcal{A}_e$ has to be reasonably complete, otherwise, no security guarantees can be given about the resulting embedding algorithm. In this contribution, $\mathcal{A}_e$ contains all detectors having the architecture of XU-Net [41] and/or SRNet [43]. The current limitation is the feasibility of the ADV-EMB attack, which can attack only differentiable detectors. However, very general approaches [19], [51] can be used to alleviate this limitation at the expense of computational complexity. Another approach would be to train differentiable surrogates, but any investigation in this direction is currently outside of the scope of this contribution.

3.4.3 *Operational embedding algorithm*



(a) Iteration $k = 0$ (b) Iteration $k = 1$ (c) Iteration $k = 2$ (d) Iteration $k = 3$

Figure 3.9: Initialization and the three first iterations of the algorithm with only one stego image. For simplification of representation, we assume that Euclidean distances (the dashed lines) between one adversarial image $\mathbf{z}$ and the boundary of the classifier f^j represents the soft output of the classifier $f^j(\mathbf{z})$. The gray lines with bigger dash represent positive distance (so when the image is in the stego region), and smaller dash represent negative distances (when the image is in the cover region). For iteration k, all values $f^j(\mathbf{z}^i)$ (for $0 \leq i \leq k$ and $0 \leq j \leq k-1$) are computed, in order to select the stego $\mathbf{y}^k$ from $\{\mathbf{z}^0, .., \mathbf{z}^k\}$ according to the min max strategy. Then f^k (in grey shade) is trained to discriminate $\mathbf{y}^k$ from cover images.

An operational version of the algorithm that relies on ADV-EMB is given by Algorithm 3 (in this case, a single convnet architecture is used to train classifiers and calibration can be omitted). The first few steps of the algorithm are illustrated in Figure 3.9. One can observe how classifiers in $\mathcal{F}^k$ surrounds the distribution of cover images and thereby restricting the choice of embedding algorithms.

Data: $\mathcal{Z}^0$ initial stego base, $\mathcal{X} = \left\{\mathbf{x}_{(1)}, .., \mathbf{x}_{(N)}\right\}$ cover base, set of detector $\mathcal{F}^0 = \left\{f^0\right\}$, $k_{\max}$
$k \leftarrow 1$;
while $k \leq k_{\max}$ **do**
  Obtain adversarial base $\mathcal{Z}^k = \left\{\mathbf{z}^k_{(1)}, .., \mathbf{z}^k_{(N)}\right\}$ where $\mathbf{z}^k_{(n)} = \text{ADV-EMB}\left(\mathbf{x}_{(n)}, f_{k-1}\right)$;
  Create stego base $\mathcal{Y}^k = \left\{\mathbf{y}^k_{(1)}, .., \mathbf{y}^k_{(N)}\right\}$ to be least detectable with respect to detectors in $\mathcal{F}^{k-1}$, $\mathbf{y}^k_{(n)} = \underset{\mathbf{z} \in \{\mathbf{z}^0_{(n)}, .., \mathbf{z}^k_{(n)}\}}{\arg\min} \underset{f \in \mathcal{F}^{k-1}}{\max} f(\mathbf{z})$;
  Train a new classifier f^k to discriminate $\mathcal{X}$ from $\mathcal{Y}^k$;
  $\mathcal{F}^k = \mathcal{F}^{k-1} \cup \{f^k_{\text{det}}\}$;
  $k \leftarrow k + 1$.
end
Return stego base $\mathcal{Y}^{k_{\max}}$

Algorithm 3: Operational algorithm executed by Alice to generate a stego base.

3.5 Experimental settings

This section details the choices of $\mathcal{A}_a$ and $\mathcal{A}_e$ used in the experiments below, together with other essential details such as calibration of classifier scores, database of images, etc.

3.5.1 *Steganographic detectors*

Since state-of-the-art steganalysis detectors are based on Convolutional Neural Networks (CNNs) [59]–[61] it should not be surprising that they are used here as well. These architectures were presented in

the previous chapter (see Section 2.4), but the readers are also referred to [56] for a general introduction and to [41], [59], [60] for their uses in steganography.

For the purpose of this work, it is sufficient to view neural networks as an efficient procedure selecting f from a large class of functions $\mathcal{F}$ minimizing the empirical error:

$$\hat{P}_{\text{err}}(f;\mathcal{X},\mathcal{Y}) = \frac{1}{|\mathcal{X}|}\sum_{\mathbf{x}\in\mathcal{X}} \mathbb{1}\left\{f(\mathbf{x}) < \tau\right\} + \frac{1}{|\mathcal{Y}|}\sum_{\mathbf{x}\in\mathcal{Y}} \mathbb{1}\left\{f(\mathbf{x}) > \tau\right\}. \quad (3.5.1)$$

An essential property of CNNs for ADV-EMB attack is their differentiability, which means that a gradient $\frac{\partial f}{\partial x}$ with respect to their inputs exists for almost every $\mathbf{x}$ and for every $f \in \mathcal{F}$.

The set of classifiers $\mathcal{F}$ is also the set of Eve's actions $\mathcal{A}_e$ and is equal to all convolutional neural networks with a given set of architectures (here XU-Net [41] or SRNet [43]).

3.5.2 *Calibrating classifier's output*

As mentioned above, the space of classifiers $\mathcal{A}_e$ can contain CNNs of different architectures and even classifiers based on a very different paradigm. In these cases, it is crucial to make their output comparable, such that min max selection in step 1 of each iteration (Equation (3.2.3)) compares meaningful quantities. A situation is illustrated in the top row in Figure 3.10 showing histograms of outputs of two classifiers on cover and stego images. Clearly, the left tail of the empirical distribution on stego images of Classifier B (denoted by $f_B(\mathcal{Y})$ for simplicity) is more spread than that of Classifier A, which means that the inner maximization in Equation (3.2.3) would prefer Classifier A over the Classifier B, although the latter is more precise.

We therefore propose to calibrate the output of a detector f by its empirical distribution function $\hat{F} : [0,1] \to [0,1]$ estimated on cover images as

$$\hat{F}(t) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{[f(\mathbf{x}_{(i)}),1]}(t),$$

where $\{\mathbf{x}_{(i)}\}_{i=1}^{N}$ are cover images. The calibrated detector, denoted $\hat{f}$, and is then defined as a composition

$$\hat{f}(\mathbf{x}) = \hat{F}(f(\mathbf{x})).$$

The effect of the calibration is shown in the bottom row in Figure 3.10. We can see that after the calibration, Classifier B would be selected by min max strategy as desired.

3.5.3 *Other implementation and experimental settings*

Embedding The embedding algorithm used to initialise the algorithm and to calculate default costs for changing elements is J-Uniward [25]

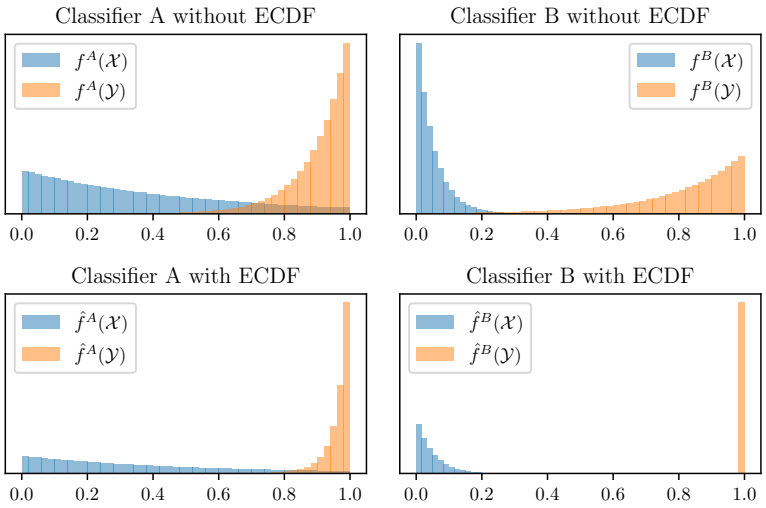


Figure 3.10: Effect of calibration allowing to make the output of two classifiers comparable on the same data base for the min max strategy.

or UERD [29]. The experiments use the JPEG version of the popular BossBase database [62] of size 512×512 in grayscale format and compressed with Quality Factor (QF) 75 or 95. Unless stated otherwise, all images are embedded using an embedding rate of 0.4 bits per non-zero AC DCT coefficient (bpnzAC) at each algorithm iteration.

Classification/Steganalysis Our implementations of XU-Net and SRNet use TensorFlow [63] library. In each iteration of the algorithm, a new steganalyzer f^k is trained by classifying cover objects $\mathcal{X}$ and stego objects $\mathcal{Y}^k$ given at the second step of the loop of Algorithm 3. Those classifiers are trained using full-size images of 512×512 DCT coefficients, 2×4000 Cover and Stego objects for training, 2×1000 for validation set and using remaining 2×5000 to estimate error rates. The training database is shuffled after each epoch. In each batch, we apply data augmentation based on random mirroring and rotation of the batch images by 90 degrees. 280 epochs are used for training using ADAM optimization algorithm. The configuration achieving the best validation accuracy is used as the result of training. For XU-Net, the classifier is trained starting with randomly initialised weights (zero-mean Gaussian with standard deviation 0.01), initial learning rate is set to 0.001 and decreased after each 5000 steps to 0.9 times the current value. The remaining parameters of ADAM are kept to the default setting. The size of mini-batch is 32 (16 cover-stego pairs). The configuration of SRNet is the one proposed in the paper [43], except the training, which lasts for 280 epochs. The size of mini-batch is 16 (8 cover-stego pairs). The experiments were run on an Nvidia GPU Quadro P6000 (24 GB of memory). Training XU-Net takes approximately 20 hours at each iteration k, SRNet 30 hours, and the generation of an adversarial database 5 hours multi-threaded on 36 cores.

Attack The ADV-EMB attack adjusting costs of DCT coefficients is implemented as described in section 3.1. Because XU-Net / SRNet uses a spatial image without rounding as input to compute partial derivatives $\frac{\partial f}{\partial x_i}$ with respect to the i^{th}-DCT coefficient, IDCT is treated as an additional layer placed before the first layer of XU-Net / SRNet. The partial derivative is consequently handled by automatic differentiation using the function `tf.gradient()` from the TensorFlow library and differentiating with respect to the image coded in the JPEG domain.

Since there is a possibility that embedding using ADV-EMB fails for some images, which means that even when we modify all costs ρ_i^+, ρ_i^- of all DCT coefficients, the corresponding stego image is classified as stego. As suggested in [50], in this case, the costs are all set to their current values without any modification, which corresponds to setting $\beta = 0$ in ADV-EMB.

3.6 Experimental comparison to prior art

This section summarises an extensive experimental study of the algorithm's properties and comparison to the prior art. First, the convergence of the algorithm is studied when the set of assumed and real Eve's detectors $\tilde{\mathcal{A}}_e$ and $\mathcal{A}_e$ are the same and when they are different. Then the proposed algorithm is compared to the prior art: the min max strategy is compared to "last iteration" and "random iteration" of [50].

The steganalytic detectors used by *real Eve* inludes XU-Net, SRNet, and linear classifiers [64] with DCTR [33] and GFR [34] feature sets. The reported error is probability under equal priors, $P_{\text{err}} = \min_{\text{Pr}_{\text{FA}}} \frac{1}{2}(\text{Pr}_{\text{FA}} + \text{Pr}_{\text{MD}})$, with Pr_{FA} and Pr_{MD} standing for the false-alarm and missed detection empirical probabilities.

Unless said otherwise, reported error rates always follows Kerkhoff's principle, which means that the detector is always trained after Alice publishes her embedding algorithm.

3.6.1 *Results*

Error rate P_{err} of XU-Net detector for eight iterations of the algorithm when Alice assumes that Eve will use XU-Net as detector is shown in the top row of Figure 3.11. Errors are shown for different payloads and different quality factors. The algorithm succeeds at significantly increasing the security in all cases. For example it makes the stego-images with payload 0.4 bpnzAC in JPEGs with QF 95 undetectable by XU-Net. The undetectability was not reached within eight iterations for other cases, but the improvement in security is still huge. Notice that the error is not strictly monotonically improving, which we attribute to (i) the training of detectors does not reach the global minimum and (ii) the ADV-EMB attack might not succeed in avoiding all detectors — a phenomenon described in more details below in sections 3.7.2 and 3.8.

The most interesting case occurs when the detectors assumed by Alice and those actually used by Eve differ as to if models used by

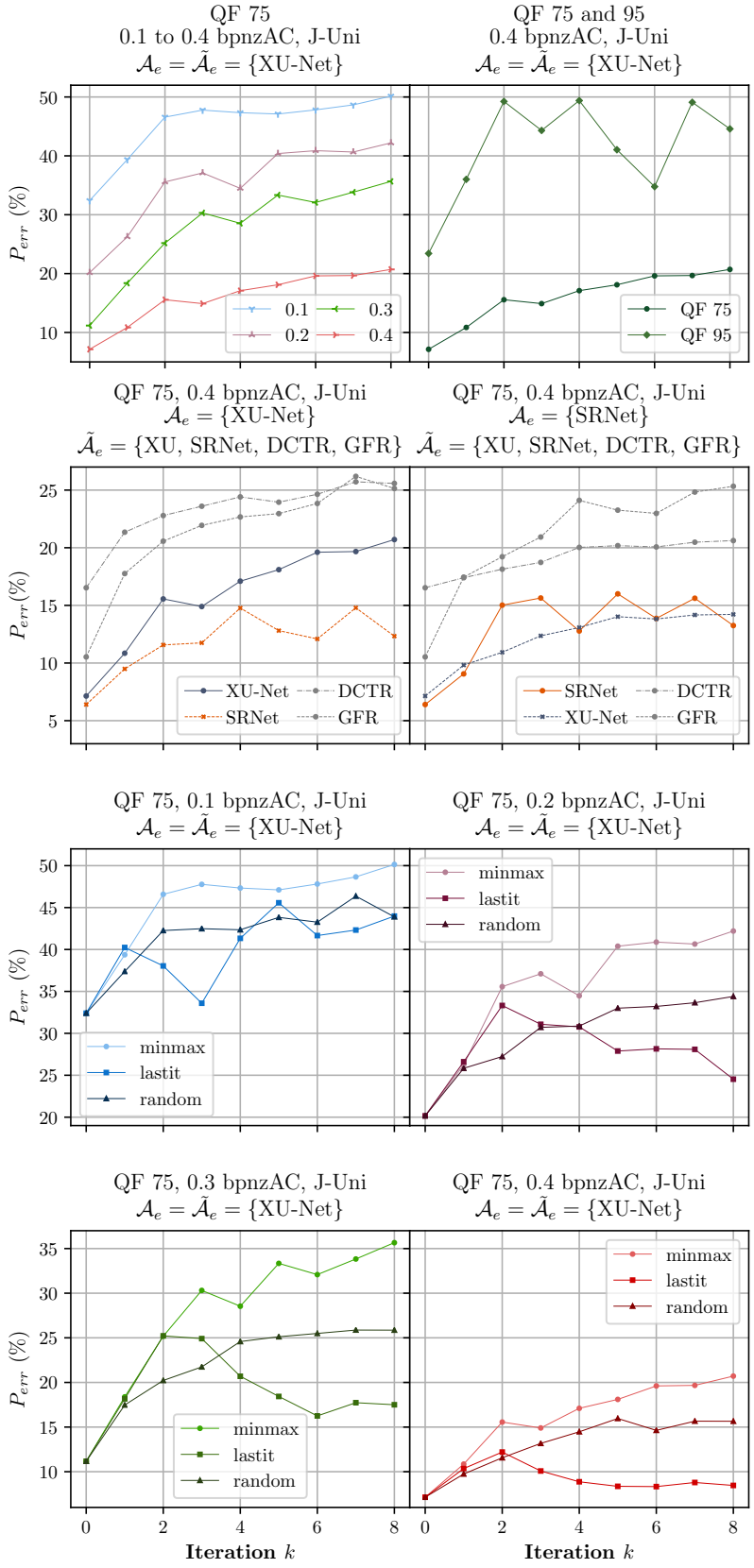


Figure 3.11: (Top row) Evolution of the algorithm for an initialization of costs with J-Uniward, $\mathcal{A}_e = \tilde{\mathcal{A}}_e = \{\text{XU-Net}\}$ and (top left) QF75, and for four different embedding rates (0.1 to 0.4 bpnzAC) ; or (top right) 0.4 bpnzAC and two different quality factor (QF 75 and 95). (Second row) Two experiments with QF 75, 0.4 bpnzAC, J-Uniward, $\tilde{\mathcal{A}}_e = \{\text{XU-Net, SRNet, DCTR, GFR}\}$ but where (left) $\mathcal{A}_e = \{\text{XU-Net}\}$ and (right) $\mathcal{A}_e = \{\text{SRNet}\}$ (Two bottom rows) Four experiments for QF 75, J-Uniward, $\mathcal{A}_e = \tilde{\mathcal{A}}_e = \{\text{XU-Net}\}$ and three strategies min max, last iteration and random, for each 0.1, 0.2, 0.3 and 0.4 bpnzAC embedding rates.

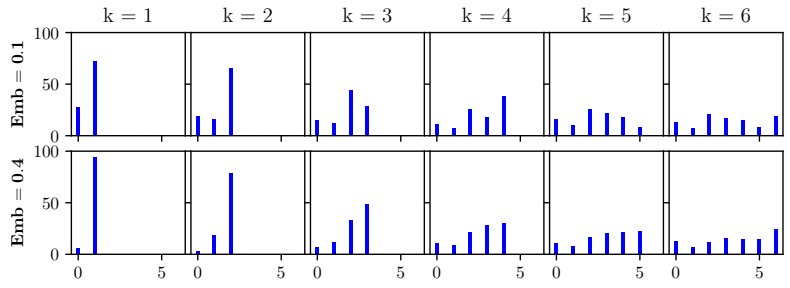


Figure 3.12: Evolution of composition of the stego data set $\mathcal{Y}^k$ over min max strategy iterations k for two experiments : classifier XU-Net, QF 75, initialization of costs with J-Uniward, and for an embedding rate of 0.3 or 0.4 bpnzAC. For each plot, bars give the proportion (in %) of images taken from $\mathcal{Z}^i$ $(0 \leq i \leq k)$ to generate $\mathcal{Y}^k$ (where i is on the x-axis).

Alice are not sufficiently rich, she might anticipate a detectable embedding. The middle row in Figure 3.11 again shows the error of XU-Net, SRNet, DCTR, GFR classifiers after the first eight iterations of the algorithm when Alice assumes XU-Net (left) or SRNet (right) in her optimization. Notably, the algorithm still improves the security even in case of a mismatch. We assume that this is because both XU-net and SRNet are sufficiently rich models.

The two bottom rows in Figure 3.11 compare the proposed algorithm to "last" and "random" strategies proposed in [50]. We see that the proposed algorithm is markedly better than both prior art solutions. It should not be surprising as, unlike them, it directly optimizes undetectability measured by Kerckhoffs' principle.

Figure 3.12 shows histograms of the iteration at which attacked detector of each stego image of Alice's stego-sets was created. This distribution is very far from the "last" strategy, which would contain a single peak at $k-1$ for iteration k. The distribution is more like a "random" strategy, which should be uniform on $\{0, 1, 2, \ldots, k-1\}$ at iteration k. The reason why the proposed algorithm is more secure than the "random" strategy is that stego images are not selected randomly, but deterministically according to min max criterion.

The security (P_{err} of classifiers) of the algorithm when the ADV-EMB attack is initialised with UERD costs and $\mathcal{A}_e$ is learned using the XU-Net architecture is shown in Figure 3.13. The security is similar to that achieved when the ADV-EMB is initialised with J-Uniward costs. It improves in the case of mismatch between assumed and real detectors (left figure), and it also improves over the prior art (right figure).

3.6.2 *2D plot of ADV-EMB*

It can be interesting to visualise the cover and stego dataset at different iterations, like proposed in Figure 3.9. To do so, we propose in Figure 3.14 to represent images in the 2D place by defining its coordinates by the raw logit of output stego class (before softmax step). We can observe that blue and orange points (respectively for cover and stego produced with J-Uniward) seem in majority correctly labelled by the three classifiers. On the other hand, $\mathcal{Y}^1$, as expected, is undetected by f^0, but detected by f^1 (after retraining). We can see that f^2, trained

Normally, we can not trace a decision boundary from only a single class logit. However, for the simplicity of visualisation (because logits give sparse clouds), we assume that the decision boundary is given by threshold 0 (dashed grey line).

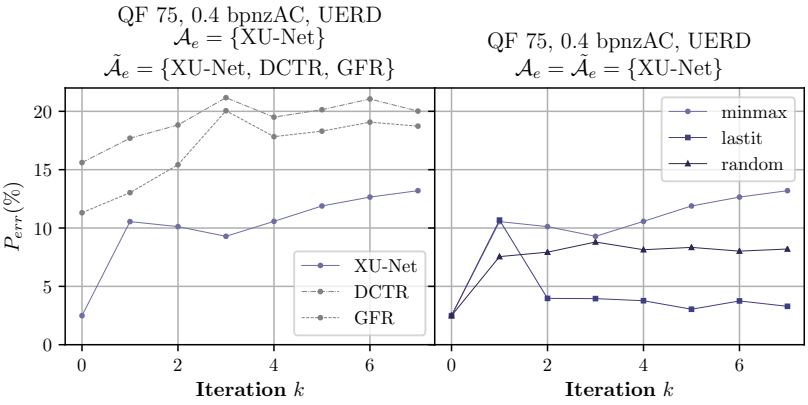


Figure 3.13: Evolution of P_{err} for XU-Net, an embedding rate of 0.4 bpnzac, QF 75, for an initialization of costs with UERD. (Left) Evolution of P_{err} of the min max strategy and of two blind steganalyzers based on GFR and DCTR features. (Right) Evolution for the 3 strategies min max, last iteration and random.

to detect $\mathcal{Y}^2$, also well detects $\mathcal{Y}^1$. It is possible that $\mathcal{Y}^1$ and $\mathcal{Y}^2$ share some images (because of the min max step).

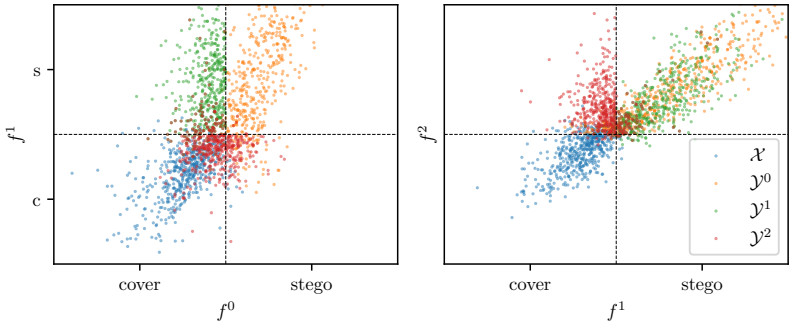


Figure 3.14: Plots showing clouds of sets of cover $\mathcal{X}$ (blue), adversarial stegos obtained with ADV-EMB at first three iterations $\mathcal{Y}^0$, $\mathcal{Y}^1$, $\mathcal{Y}^2$ (respectively orange, green and red).

3.7 Optimizing against more architectures

3.7.1 *Performance analysis*

In the previous section, the set of classifiers $\mathcal{A}_e$ used by Alice contained neural networks with the same architecture differing only in weights. But as was repeatedly emphasised, the set $\mathcal{A}_e$ should be as complete as possible, which means CNNs with different architectures. Below, $\mathcal{A}_e$ contains all neural networks with XU-Net and SRNet architectures. This was achieved by extending the set $\mathcal{F}^k$ in each iteration by two networks, one with XU-Net and one with SRNet architecture. Outputs of these detectors are always calibrated as was described above in Sectionß 3.5.2. Otherwise, the experimental settings and the algorithm are unchanged.

The error of four steganalyzers (DCTR and GFR are not in $\mathcal{A}_e$) for the first eight iterations when the algorithm optimized embedding message with payload 0.4 bpnzAC in JPEG images with QF 75 is shown in Figure 3.15. The behaviour is similar as observed above as the algorithm improves the undetectability significantly. Table 3.1 summarises the increase of the undetectability (error rate) against J-

Uniward as measured by different classifiers (in rows) when $\mathcal{A}_e$ contains either XU-Net, SRNet, or both (in three last columns). In line with theoretical expectations, Algorithm 3 with $\mathcal{A}_e$ containing both architectures achieves highest undetectability (minus noise) with respect to all four tested detectors. Specifically, the detectability by SRNet presently considered the most powerful detector jumps from 6.3% to 16.3%, which is almost a three-fold improvement in security. This also means that the proposed algorithm delivers the most secure steganographic algorithm at the time of writing.

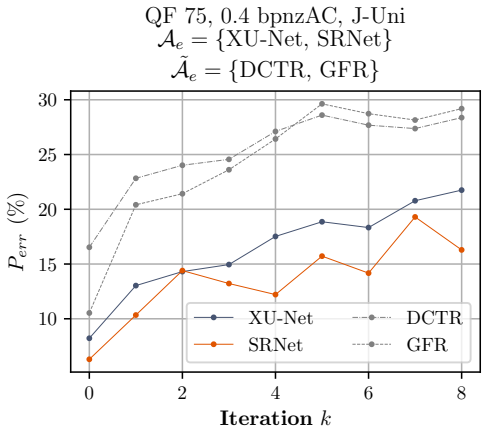


Figure 3.15: Evolution of P_{err} for the experiment with double adversary : where QF 75, 0.4 bpnzAC, J-Uniward, and where there are two assumed steganalyzers : XU-Net and SRNet, and with two more real steganalyzers based on DCTR and GFR features.

$\tilde{\mathcal{A}}_e$	J-Uniward	$\mathcal{A}_e$ (for QF 75, 0.4 bpnzac, J-Uniward)		
		{XU-Net}	**{SRNet}**	**{XU-Net, SRNet}**
{XU-Net}	7.1%	+ **13.6%**	+ 7.1%	+ **13.5%**
{SRNet}	6.3%	+ 5.9%	+ **6.9%**	+ **10.0%**
{DCTR}	16.5%	+ 9.1%	+ 4.1%	+ 11.8%
{GFR}	10.5%	+ 14.6%	+ 14.8%	+ 18.7%

Table 3.1: The first column shows the baseline detectability of J-Uniward : it gives the error rate of four types of detectors (in rows). Then, the next three columns show the gain in error rate $P_{\text{err}}(k=8) - P_{\text{err}}(k=0)$ for three experiments, QF 75 and embedding rate of 0.4 bpnzAC, J-Uniward, and with min max strategy. First with $\mathcal{A}_e$ = {XU-Net}, second with $\mathcal{A}_e$ = {SRNet} and third with $\mathcal{A}_e$ = {XU-Net, SRNet} (so with double adversaries). Bold: evolution of error rate when there is a match between $\mathcal{A}_e$ and $\tilde{\mathcal{A}}_e$ in the experiment. For each column, non-bold results are for mismatches.

3.7.2 *Compositions of training sets*

Figure 3.15 showing P_{err} of all steganalyzers with respect to iteration on the algorithm suggests that Alice should be sending stego-images created by ADV-EMB attacking SRNet, as these detectors produce the lowest error. Figure 3.16 shows distribution of algorithms used to create stego-images for each iteration. Surprisingly, even though SRNet has a lower error rate, stego-images are consistently created by attacking XU-Net, which is everything but intuitive.

We believe that this problem stems from the weakness of ADV-EMB attack, which calculates gradients of detectors outputs only once during embedding (see details in section 3.1). It can lead to cases when trying to evade one classifier (e.g. SRNet), the image is detectable by a different classifier (e.g. XU-Net). It is measured in terms of *transferability* of attacks, defined as

- $T^k_{XU} = P\big(f^{k-1}_{SR}(\mathbf{z}^k_{XU}) < 0.5 \big| f^{k-1}_{XU}(\mathbf{z}^k_{XU}) < 0.5\big)$,

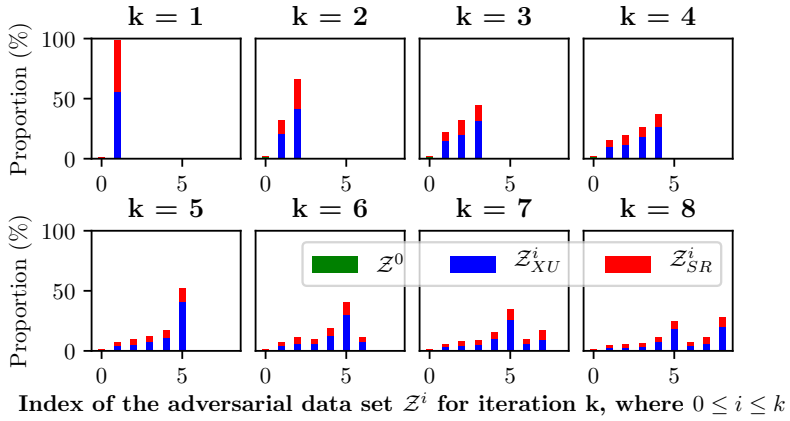


Figure 3.16: Composition of $\mathcal{Y}^k$ set with two steganalyzers, an embedding rate of 0.4 bpnzac, QF 75. Green bar represent the proportion of stego chosen among $\mathcal{Z}^0$ (so with the costs of J-Uniward). For $i > 0$, blue (resp. red) bars represent the proportion of stego chosen among $\mathcal{Z}^i_{XU}$ (resp. $\mathcal{Z}^i_{SR}$), i.e. the adversarial stego contents attacking f^{i-1}_{XU} (resp.f^{i-1}_{SR}).

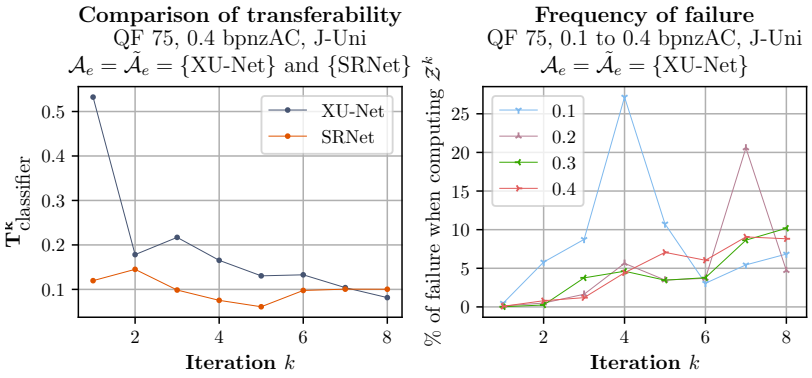


Figure 3.17: (Left) Evolution of the transferability T^k_{XU}, T^k_{SR} over iteration k of adversarial data bases $\mathcal{Z}^k_{XU}$, $\mathcal{Z}^k_{SR}$. (Right) Evolution of the frequency of failure for multiple embedding rates

- $T^k_{SR} = P\big(f^{k-1}_{XU}(\mathbf{z}^k_{SR}) < 0.5 \big| f^{k-1}_{SR}(\mathbf{z}^k_{SR}) < 0.5\big)$,

which expresses the probability that stego-images created by ADV-EMB against XU-Net will be undetectable by SRNet and vice versa. Transferability shown in Figure 3.17 for each iteration is generally very low, but that of ADV-EMB attacking XU-Net is generally higher than that of attacking SRNet. This implies that ADV-EMB has to be somehow adapted to attack a set of classifiers instead of just one, but this work is clearly outside of the scope of this contribution. Figure 3.17 shows the probability of failure of ADV-EMB attack, which generally increases as the algorithm progresses, as the distribution of cover images (and its support) is better captured, as illustrated in Figure 3.9.

3.8 Note on the initialization of CNNs

Ref. [65] introduced the concept of Curriculum Learning (CL), which is a technique used to improve the learning of a classifier for low payloads or low-quality factors by training them on more manageable problems. We have experimented with this approach by initialising learning of a classifier at iteration k with parameters of the classifier trained at the previous iteration $k-1$. Alternatively and as used in all experiments above, the classifiers were initialised entirely at random.

Figure 3.18 shows error rate of both algorithms when $\mathcal{A}_e$ contains only XU-Net detectors. The experiment used JPEG images with QF 75, 0.4 bpnzAC, and costs in ADV-EMB were initialised by J-Uniward. The experimental results show that the algorithm where curriculum learning is not used achieved higher accuracy of detection than the one using it. We believe that the detectors with curriculum learning might be stuck in a suboptimal local minimum.

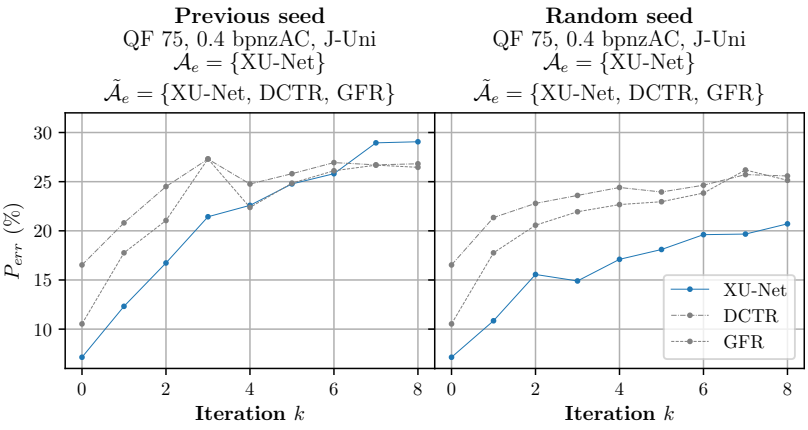


Figure 3.18: Effect of the initialization of CNN on the P_{err} of the algorithm (with XU-Net, initialization of costs with J-Uniward, QF75, min max strategy and an embedding rate of 0.4 bpnzAC). (Left) f^k was seeded by f^{k-1}, and (right) f^k is randomly seeded.

3.9 Flaws of ADV-EMB

But the algorithm ADV-EMB has weaknesses. First, we can notice that the ratio β, controlling the size of the adjustable group so how *adversarial* is the attack, is increasing with iterations, and the attacks fail at some point as shows Figure 3.19. We recall the ratio β of ADV-EMB is the smallest ratio of modified coefficients among all coefficients such as the stego is misclassified by the previous classifier. For $k = 1$, the attack is easy, and a low ratio β of costs are updated to defeat f^0. But for further iterations, a bigger and bigger β is required, and there are more and more failures.

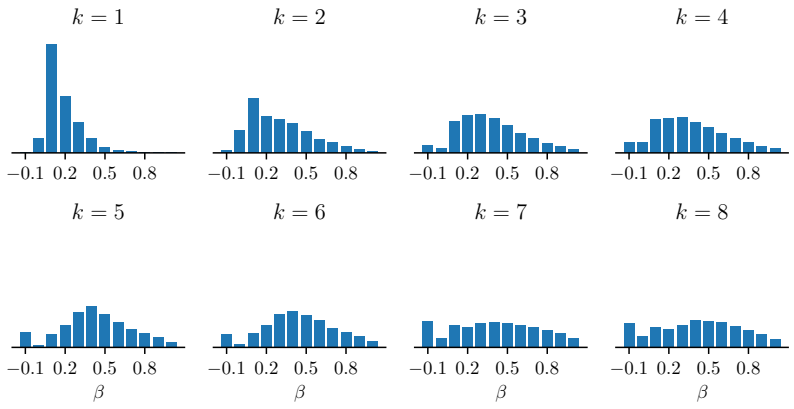


Figure 3.19: Histograms of value of β needed to produce $\mathcal{Z}^k$, i.e. the adversarial stego version $\mathbf{z}^k$ of $\mathbf{x}$ such as $f^{k-1}(\mathbf{z}^k) < 0.5$. Results plotted from the experiment with images at QF 75, J-Uniward and payload of 0.4 bpnzAC, from $k = 1$ to 8. The bar for $\beta = -0.1$ is for the quantity of failures, meaning that it counts how many images are classified as stego with a $\beta = 1$ (meaning all cost are changed and the image still doesn't fool f^k).

We can evaluate the efficiency of the attack by measuring the value

of the minimum reached during Alice's move in the protocol given by Algorithm 3. Figure 3.20 shows, w.r.t. iteration k the evolution of the average output of classifier f^k evaluated over all stego images in the database of $\mathbf{y}^k_{(n)}, n \in [1, N]$ where we recall that $\mathbf{y}^k_{(n)}$ is defined as follows:

$$\mathbf{y}^k_{(n)} = \underset{\mathbf{z}\in\{\mathbf{z}^0_{(n)},\dots,\mathbf{z}^k_{(n)}\}}{\arg\min} \max_{f\in\mathcal{F}^{k-1}} f(\mathbf{z}). \tag{3.9.1}$$

Note that because $f : \mathcal{I} \to [0, 1]$, $\frac{1}{N}\sum_{n\in[1,N]}\left[f(\mathbf{y}^k_{(n)})\right] \leq 1$. For high payload (for example 0.4), you can observe that the min max value is reaching the maximum value 1.

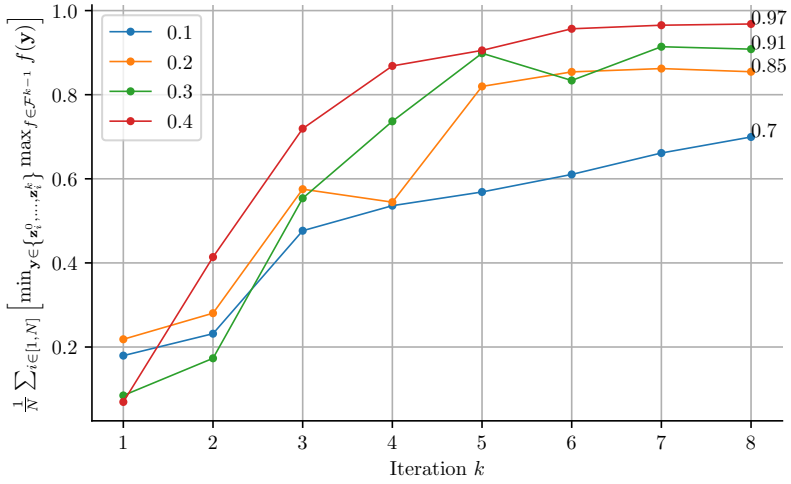


Figure 3.20: Evolution of the value of $\frac{1}{N}\sum_{n\in[1,N]}\left[f(\mathbf{y}^k_{(n)})\right]$ w.r.t. iteration k for protocols with JPEG images at QF 75, with J-Uniward and XU-Net, and for four payloads from 0.1 to 0.4 bpnzAC (4 colored lines).

The value of min max increasing close to 1 is compatible with the high error rate of classification given in the protocol. For protocol at 0.4 bpnzAC, XU-Net at iteration 8 gives an error rate of 22.0%. But we show on Figure 3.21 that any classifier f^k gives an error rate below 25.6% to classify the whole set of stego obtained via Equation 3.9.1 at iteration 8.

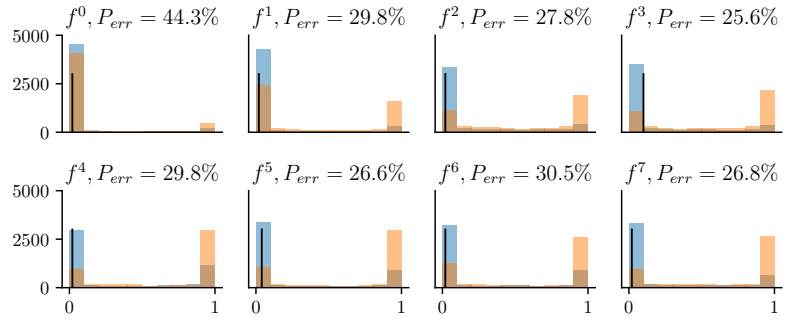


Figure 3.21: Histograms of stego class probability given by different classifier f^k, $k \in \{0, \dots, 7\}$ evaluated on cover images (blue bars) and stego images $\mathbf{y}^8_{(n)}$ (orange bars) obtained via Equation 3.9.1 for protocol at QF 75, J-Uniward, XU-Net at payload 0.4 bpnzAC.

Finally, observation from Figure 3.20 means that the solving of equation min max becomes more and more difficult, as the values

reache nearly the maximal value 1 (0.97 for payload 0.4). It can be explained by the following observation. The general protocol in Algorithm 2 suggests to Alice to solve

$$h_{\text{emb}} = \underset{h_{\text{emb}} \in \mathcal{A}_a}{\arg\min} \max_{f \in \mathcal{F}^{k-1}} f(h_{\text{emb}}(\mathbf{x})).$$

But our practical algorithm proceeds in two steps: (i) compute $\mathbf{z}^k_{(n)} = \text{ADV-EMB}\left(\mathbf{x}_{(n)}, f_{k-1}\right)$ (adversarial image defeating f_{k-1}) then (ii) pick $\mathbf{y}^k_{(n)} = \underset{\mathbf{z} \in \{\mathbf{z}^0_{(n)}, \dots, \mathbf{z}^k_{(n)}\}}{\arg\min} \max_{f \in \mathcal{F}^{k-1}} f\left(\mathbf{z}\right)$.

It produces a terrible flaw because $\mathbf{z}^k$ produced by ADV-EMB aims only at fooling f^{k-1} and might be detectable by other classifiers, such as the min max value could be higher than hoped. We show in Figure 3.22 that other classifiers detect this new embedding in the set of Alice. Indeed, the blue diagonal show that at each new iteration $k \geq 1$, $\mathcal{Z}^k$ fools only f^{k-1}. However, with the red squares below the blue diagonal, we can see that the new database does not fool other classifiers $f^l, 1 \leq l < k-1$.

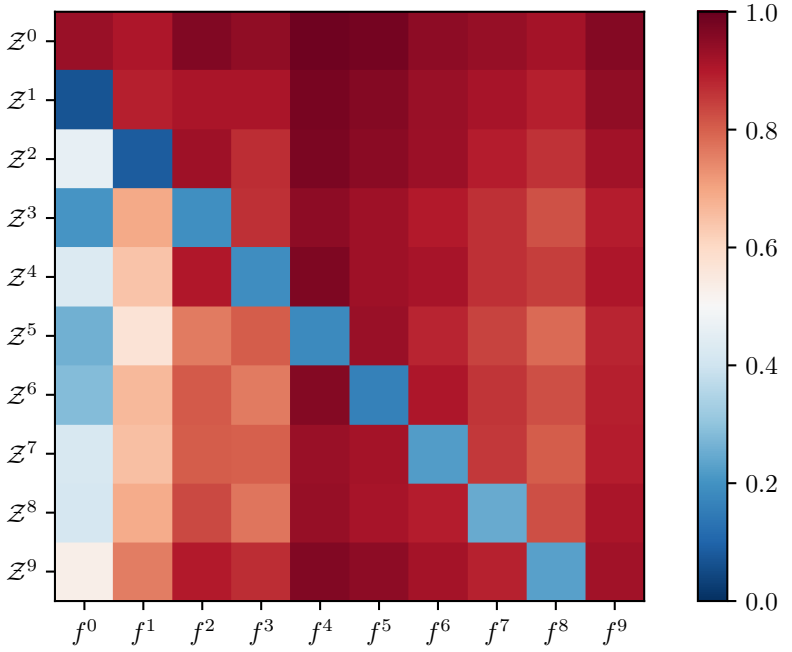


Figure 3.22: For the protocol with JPEG images at QF 75 and payload 0.4 bpnzAC, with J-Uniward and XU-Net, average output of each classifier f^j (columns) evaluated on each adversarial stego database $\mathcal{Z}^i$ (rows). Blue color is for images detected as cover (the probability of stego class is below 0.5), whereas red if for images classified as stego.

3.10 Conclusions of this chapter

This contribution builds a min max optimization upon the Kerckhoffs' principle. Since direct optimization of this criterion is computationally infeasible, we simplify the optimization problem by giving Eve an unrealistic advantage — she can choose her detector after she observes Alice's image. We advocate this simplification to be fair, as it is used exclusively by Alice during optimization of her steganographic scheme, and the evaluation of the security of the resulting algorithm is fair and does as is standard within the field.

Although the proposed algorithm is general, the realisation used in this contribution relied on two recent innovations: convolutional neural networks implementing a general class of steganographic detectors and adversarial embedding capable of embedding a message while being undetectable by a given detector. The extensive experimental results demonstrate the superiority of the proposed algorithm with respect to the prior art in the JPEG domain. Specifically, the most secure version increases the error rate of classification of JPEGs with embedded message of size 0.4 bpnzAC by SRNet by 10% comparing to J-Uniward: it jumps from 6.3% to 16.3%. This increase in security should not be surprising since the presented algorithm automatically plays the game, which the community plays implicitly since the birth of the field.

The purpose of our second contribution is to improve Alice's adaptative steganographic method to bypass a detector by building a method with fewer heuristics.

Differentiable steganography with Backpack

4

4.1 Introduction

We saw that state of the art steganographic algorithms rely on the principle of minimising a distortion function, which needs to be additive for practical reasons. Before hiding the message (embedding), a steganographic algorithm assigns an embedding cost to each modifiable coefficient (e.g. a pixel or DCT coefficient if the cover object is an image). The message is then hidden by syndrome trellis codes [66], which minimises the additive distortion function under the constraint of communicating a message. For research purposes, the act of embedding a concrete message is skipped in favour of a mere simulation [67], [68]. Presently, the design of new steganographic algorithms, therefore, boils down to defining the embedding costs.

An evolution, compared to heuristic, is to automate the design of detection functions by constructing stegnalyzers iteratively via a min max protocol. It does not use the fixed estimator of embedding changes; instead, the distortions associated with each DCT sample are optimised through adversarial embedding (ADV-EMB) [31] for a given image and steganalyst. The set of steganalysts created by the min max protocol plays the role of non-additive distortion functions, as they measure the detectability of a given image. Indeed the adversaries (here the classifiers) capture mid-range dependencies between DCT coefficients which are by definition non-additive. In order to overcome this issue, the embedding can induce correlations by either using lattices (like ADV-EMB in sections 1.5.10) and/or by crafting specific non-symmetric additive costs as done here.

This chapter proposes a method called *Backpack* (for BACKPropagable AttaCK)[1] fixing flaws of ADV-EMB by finding embedding costs by gradient descent with constraint on message length included by means of implicit differentiation. To put the proposed method into a broader context, it optimises parameters of additive distortion function (embedding costs) by challenging a non-additive distortion function (a set of steganalyzers) for a given image and message length. In this sense, the method is general. The experimental evaluation demonstrates its advantages compared to ADV-EMB by improving the security of a steganographic algorithm found by min max protocol [2] by 11% when the distortion function is a deep neural network with a Xu-Net [41] architecture.

This contribution is organised as follows. The next section 4.2 recalls important prior arts for explaining how from a cost map, we can

[1] The name is derived from methods which inspired the work: backpropagation and adversarial attacks on neural networks.

simulate the optimal embedding of a message and define the problem. Section 4.3 shows step by step how to calculate the gradient of a differentiable steganalyzer with respect to embedding costs. Experimental section 4.4 shows the effect of the proposed scheme on the security of the obtained embedding costs.

4.2 Background

In this section, we begin by introducing the problem; then, we recall the non-back-propagable pipeline to simulate a stego from a cost map.

4.2.1 *Problem formulation*

The problem we solve is for a given cover object $\mathbf{x}$ to find an embedding cost map $\boldsymbol{\rho}$ minimizing detectability of a stego object $\mathbf{y} = \mathbf{x} + \mathbf{b}$ by a non-additive distortion function $f(\mathbf{y})$ (read steganalyst), where $\mathbf{b} \sim P_{\mathbf{b}}(\mathbf{b}|\boldsymbol{\rho}, \lambda)$ and f being almost everywhere differentiable with respect to $\mathbf{y}$. We therefore focus on *simulation* of embedding changes. The stego object $\mathbf{y} = \mathbf{x} + \mathbf{b}$ is a realization of a random variable and we minimize the expected detectability over all possible stego objects written as :

$$\arg\min_{\boldsymbol{\rho},\lambda} \mathbb{E}_{\mathbf{b}\sim P_{\mathbf{b}}(\mathbf{b}|\boldsymbol{\rho},\lambda)}[f(\mathbf{x}+\mathbf{b})], \tag{4.2.1}$$

subject to the entropy constraint:

$$H(P_{\mathbf{b}}(\mathbf{b}|\boldsymbol{\rho}, \lambda)) = |\mathbf{m}|. \tag{4.2.2}$$

We want to tackle the above problem using a classical optimisation method to remove all heuristics. Here we propose to use the very famous gradient descent with respect to $\boldsymbol{\rho}$. However, its use for this problem is complex for two reasons: first, the optimisation problem contains an implicit constraint on the entropy; second, the gradient of the expectation of f with respect to $\boldsymbol{\rho}$ does not have an analytical expression, and its exact computation would be prohibitively expensive. To compute it exactly, one would need to sum over the support of all stego images (for a given cover), which has $|\mathcal{B}|^n$ (recall that $|\mathcal{B}|$ is the cardinality of embedding changes and n is the number of coefficients that can be modified during embedding).

We are therefore interested into finding a computable value of:

$$\nabla_{\rho}\mathbb{E}_{\mathbf{b}\sim P_{\mathbf{b}}(\mathbf{b}|\boldsymbol{\rho},\lambda)}[f(\mathbf{x}+\mathbf{b})]. \tag{4.2.3}$$

The rest of this section, therefore, focuses on an *approximation* of (4.2.3) that would be sufficiently accurate while being computationally cheap. We will first present the *forward pipeline*, which is all the successive operations made from the cost $\boldsymbol{\rho}$ to obtain a stego image $\mathbf{y}$.

4.2.2 *The forward pipeline*

It was presented in the previous chapter in section 1.4.4 that practically, steganography by cover modification can be simulated by draw-

ing a stego simulating the impact of the embedding reached by an optimal coding function (see section 1.4.4). Those steps, shown in Figure 4.1, have already been presented, but it will be recalled here following steps 1 to 5 on the Figure.

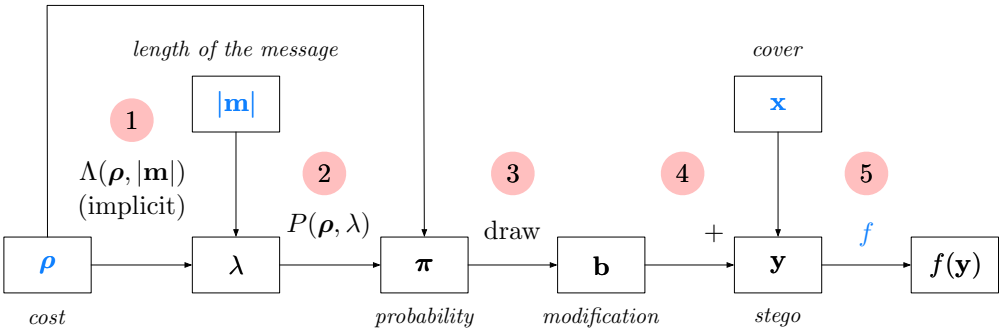


Figure 4.1: Pipeline to obtain a *simulated* stego image $\mathbf{y}$ from a cover image $\mathbf{x}$, a cost map $\boldsymbol{\rho}$ and a size of the message $|\mathbf{m}|$. Operations, labeled by number in red, are plotted by arrows. Variables are located in rectangles, and blue variables are the inputs of the pipeline. Steganalysis can be applied at the end by evaluating a detector f on the stego image.

Steps 1 and 2: computing optimal probabilities. The first two steps are intrinsically binded, it is why they are presented at the same time. For a given size of message $|\mathbf{m}|$ in bits and an additive distortion function described by its cost map $\boldsymbol{\rho}$, a theoretical results gives the distribution of stego images provided by the optimal coding function. The probability distribution of stego modifications $P_{\mathbf{b}}(\mathbf{b}|\boldsymbol{\rho}, \lambda) = \prod_{i=1}^{n} P_{b_i}(b_i = j|\boldsymbol{\rho}, \lambda)$ gives independant probability distributions P_{b_i} over the image coefficients (thanks to the additivity property of the distortion function). The probability of modification of cover coefficient indexed by i by value j is equal to

$$\pi_i^j = P_{b_i}(b_i = j|\boldsymbol{\rho}, \lambda) = \frac{e^{-\lambda\rho_i^j}}{\sum_{k\in\mathcal{B}} e^{-\lambda\rho_i^k}} = p_i^j(\boldsymbol{\rho}, \lambda). \tag{4.2.4}$$

Let's define the functions $p_i^j(\boldsymbol{\rho}, \lambda)$ which take as input the cost $\boldsymbol{\rho} = \{\rho_i^j\}_{i,j}$ and the scalar λ, and yield outputs π_i^j.

One can check that $\forall 1 \leq i \leq n$, P_{b_i} is a distribution probability. Indeed $\forall b \in \mathcal{B}$, we have $0 \leq \pi_i^j \leq 1$ and $\sum_{k\in\mathcal{B}} \pi_i^k = 1$.

Looking at its definition above, one might think that π_i^j only depends on the values $\rho_i^k, \forall k \in \mathcal{B}$. But it also depends on scalar λ tuned such that the entropy of the probability distribution $P_{\mathbf{b}}$ is equal to the length of the message, i.e

$$H(\boldsymbol{\pi}) = -\sum_{i=1}^{n}\sum_{j\in\mathcal{B}} \pi_i^j \log \pi_i^j = |\mathbf{m}|. \tag{4.2.5}$$

Finally, λ is itself a function of $\boldsymbol{\rho}$ and $|\mathbf{m}|$, because of the entropy constraint of Equation 4.2.5. But it exists no explicit expression of this function. Because we do want to emphasize on the fact that λ depends on those two variables, we will sometime write λ as $\lambda = \Lambda(\boldsymbol{\rho}, |\mathbf{m}|)$.

Solving equation $H(P_{\mathbf{b}}(\mathbf{b}|\boldsymbol{\rho}, \lambda)) = |\mathbf{m}|$ in order to find λ is usually achieved by binary search. An animation of such solving is shown on Figure 4.2.

At the end of steps 1 and 2, the pipeline gives for each coefficient indexed by i a probability distribution P_{b_i}, described by $(\pi_i^j)_{,j\in\mathcal{B}}$.

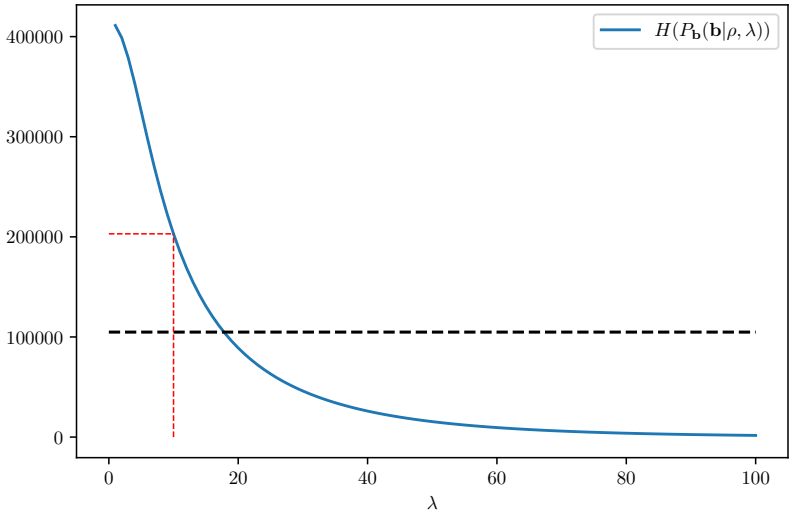


Figure 4.2: For an arbitrary cost map $\boldsymbol{\rho}$, entropy H of probabilities obtained by Equation 4.2.4 from $\boldsymbol{\rho}$ and λ, w.r.t. λ. The arbitrary target entropy $|\mathbf{m}|$ is plotted by an horizontal black dashed line. $\Lambda(\boldsymbol{\rho}, |\mathbf{m}|)$ is obtained at the end of the binary search, read at the abscissa of the red vertical line. The animation (GIF readable with Adobe Reader) shows the evolution of the value of λ during the search.

Step 3: Drawing a modification. The probability map $\boldsymbol{\pi} = (\pi_i^j)$ contains the optimal probability of modification j for each image coefficient indexed by i. It means that the modification b_i is drawn according to the probability law P_{b_i}, which can be written as

$$b_i = j \text{ with probability } \pi_i^j. \tag{4.2.6}$$

The modfications applied to each pixel can be drawn independently according to P_{b_i}. At the end, we obtain a modification map $\mathbf{b} = (b_i)_i$, where $b_i \in \mathcal{B}$.

Step 4: applying the modifications to the cover. The stego object $\mathbf{y}$ is obtained simply by term-wise summation of vector $\mathbf{b}$ to the cover vector $\mathbf{x}$, i.e. $\forall 0 \leq i \leq n, y_i = x_i + b_i$. We say that $\mathbf{y}$ is a simulated stego because it is obtained from modifications drawn according to some probability distribution (step 3). In order to facilitate the understanding of those objects, it is shown in Figure 4.3 the shape of them.

To each simulated stego corresponds a real stego coded using STC with a payload size slightly smaller than $|\mathbf{m}|$ [68].

Step 5: evaluating the stego. At the end of the pipeline, one can evaluate the stego by a detector f. This last step is related to steganalysis, but we introduce it because this section aims at providing a way to avoid a detector f.

4.2.3 *Re-parametrization trick*

The idea we propose is to optimise the cost $\boldsymbol{\rho}$ w.r.t. the detectability of a detector f by a classical method of optimisation. For example, we could use a gradient descent technique on the costs in order to decrease $f(\mathbf{y})$. But we would need to compute the gradient $\nabla_{\boldsymbol{\rho}} \mathbb{E}[f(\mathbf{y})]$, as if for a given $\boldsymbol{\rho}$, we could update it by the following formula

$$\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} - \alpha \nabla_{\rho} \mathbb{E}_{\mathbf{b} \sim P_{\mathbf{b}}(\mathbf{b}|\boldsymbol{\rho},\lambda)}[f(\mathbf{x}+\mathbf{b})], \tag{4.2.7}$$

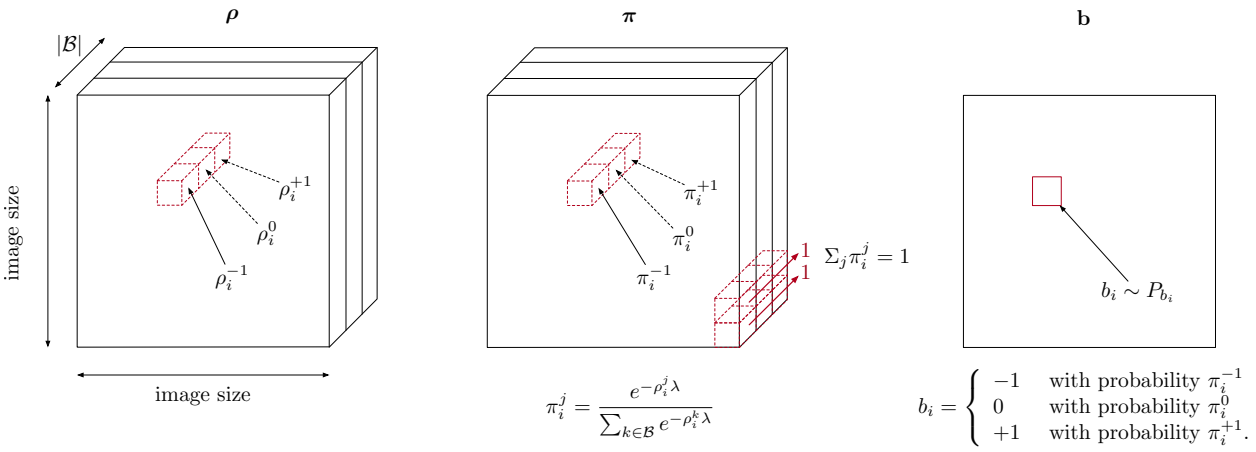


Figure 4.3: Vizualization of shapes of the cost $\boldsymbol{\rho}$, the probability $\boldsymbol{\pi}$ and the modification maps $\mathbf{b}$, in the case where $\mathcal{B} = \{-1, 0, +1\}$. Note that we illustrated an image as being two-dimensional, but an element in this 2D space is indexed by a single integer i in order to lighten the notations.

where $\alpha > 0$ is the step size of the gradient descent.

The first idea is to reparameterize the distributions P_{b_i}, such as we can compute the sample b_i as a deterministic function B of $\boldsymbol{\pi}_i = (\pi_i^j), j \in \mathcal{B}$ and a vector of independent random variables $\mathbf{r}_i$ drawn from distribution R. We have $b_i = B(\boldsymbol{\pi}_i, \mathbf{r}_i)$. The path-wise gradients from b_i to $\boldsymbol{\pi}_i$ can then be computed without encountering any stochastic nodes. This decomposition is illustrated on Figure 4.4, replacing the *draw* step (3) in initial Figure 4.1. This is a very general scheme, and many pairs of $(B, \mathbf{r})$ could fit for drawing according to a discrete distribution.

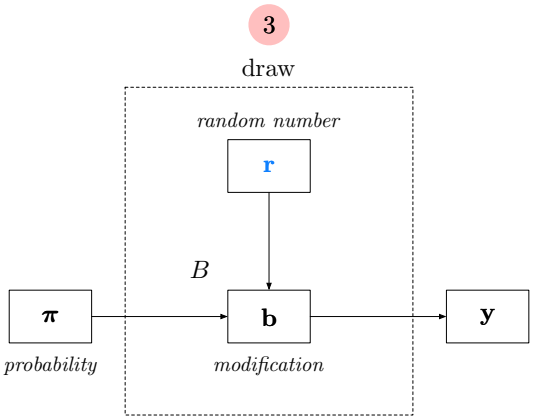


Figure 4.4: Decomposition of the *drawing* step (3) of Figure 4.1 into (i) a purely random step which is the sampling of input $\mathbf{r}$, and (ii) a deterministic function which involves the probabilities $\boldsymbol{\pi}$.

Decoupling modification probabilities from the stochasticity of the randomly drawn modifications will allow to compute the gradient of DCT coefficients y_i w.r.t. the distribution parameters π_i^j, as it is shown in section 4.3.1. This reparametrization allows also to permute the gradient and the expectancy:

$$\begin{aligned} \nabla_{\boldsymbol{\rho}} \mathop{\mathbb{E}}_{\mathbf{b} \sim P_{\mathbf{b}}(\mathbf{b}|\boldsymbol{\rho},\lambda)} [f(\mathbf{x}+\mathbf{b})] &= \nabla_{\boldsymbol{\rho}} \mathop{\mathbb{E}}_{\mathbf{r} \sim R} [f(\mathbf{x} + B(\boldsymbol{\pi}, \mathbf{r}))] \\ &= \mathop{\mathbb{E}}_{\mathbf{r} \sim R} [\nabla_{\boldsymbol{\rho}} f(\mathbf{x} + B(\boldsymbol{\pi}, \mathbf{r}))], \end{aligned} \tag{4.2.8}$$

where $\boldsymbol{\pi}$ depends on the variable $\boldsymbol{\rho}$. The main advantage of re-parametrization is that the gradient in Equation (4.2.3) can be now estimated using k

Monte Carlo samples $\mathbf{r}_1, \dots, \mathbf{r}_k$ as

$$\mathop{\mathbb{E}}_{\mathbf{r}\sim R}[\nabla_{\boldsymbol{\rho}} f(\mathbf{x} + B(\boldsymbol{\pi}, \mathbf{r}))] \approx \frac{1}{K} \sum_{j=1}^{K} \nabla_{\boldsymbol{\rho}} f(\mathbf{x} + B(\boldsymbol{\pi}, \mathbf{r}_j)). \tag{4.2.9}$$

The number K of drawn samples $\mathbf{r}_1, \dots, \mathbf{r}_K$, controls the trade-off between the variance of the estimate and the computational complexity. In experiments presented in this chapter, K was selected to the highest number which the GPU memory allowed.

Now the problem is focused on how to compute $\nabla_{\boldsymbol{\rho}} f(\mathbf{y}) = \nabla_{\boldsymbol{\rho}} f(\mathbf{x} + B(\boldsymbol{\pi}, \mathbf{r}))$, for fix random values $\mathbf{r}$. It is the subject of the next section.

4.2.4 *The backward pipeline and the associated problems*

Given the pipeline in Figure 4.1, the computation of the gradient is the backward operation from the right to the left, and could be computed by the chain rule of the steps 1 to 5. Explicity, it gives:

$$\nabla_{\boldsymbol{\rho}} f(\mathbf{y}) = \nabla_{\mathbf{y}} f(\mathbf{y}) \frac{\partial \mathbf{y}}{\partial \mathbf{b}} \frac{\partial \mathbf{b}}{\partial \boldsymbol{\pi}} \frac{d\boldsymbol{\pi}}{d\boldsymbol{\rho}}. \tag{4.2.10}$$

The last term of the chain rule is the total derivative of $\boldsymbol{\pi}$ w.r.t. $\boldsymbol{\rho}$. It is necessary because $\pi_i^j = p_i^j(\boldsymbol{\rho}, \lambda)$ is a function a two dependant variables, because $\lambda = \Lambda(\boldsymbol{\rho}, |\mathbf{m}|)$.

Given the value of the total derivative, this gradient depends of $\frac{\partial \pi}{\partial \rho}$, $\frac{\partial \pi}{\partial \lambda}$ and $\frac{\partial \lambda}{\partial \rho}$.

In this formula, there are two problems, shown in Figure 4.5.

Suppose that f is a function of two variables, x and y, and that y depends on x, we have:

$$\frac{df(x, y(x))}{dx} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial x} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial x}$$

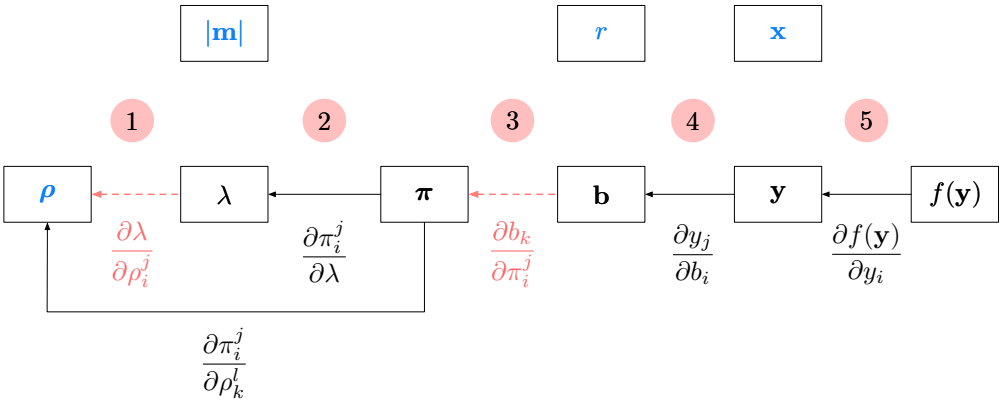


Figure 4.5: The backward pipeline to compute $\nabla_{\boldsymbol{\rho}} f(\mathbf{y})$ with the chain rule. The 1 and 3 backward steps are the two critical points for the gradient.

The considerable difficulty is the computation of the gradient of the modifications b_i with respect to the probabilities π_j^k. Because b_i is an integer drawn according to the probabilities $(\pi_i^j)_{j\in\mathcal{B}}$, there is no direct expression of the gradient.

The second issue comes from the computation of λ with respect to the cost $\boldsymbol{\rho}$. Because we saw that no explicit expression of Λ exists, the gradient is not straightforward.

Now that we understand the importance of solving those two problems, the next section, which is the core of this contribution, proposes solutions to tackle this problematic differentiation.

4.3 Differentiable steganography

To simplify the notation, the calculation of the expected gradient is introduced for a single coefficient. It means that the modification b of a coefficient is a scalar with probability distribution described by a vector $\boldsymbol{\pi}$ of length $|\mathcal{B}|$. Since embedding changes are assumed to be independent (the additivity of our distortion function constrains it), the generalisation to all coefficients of an object is straightforward.

Calculating the gradient of the expectation of a discrete probability distribution with respect to its parameters is a very well studied problem. From the vast prior art, we have chosen the method [69] relying on the Gumbel distribution. This technique has the advantage of giving a general formula to draw samples according to any discrete distribution so that it can be used without a modification for n-ary coding, and its theoretical properties are well analysed.

4.3.1 *Transforming step 3 into a differentiable step*

In order to simplify the notation, let's focus on a single image coefficient x which is changed by value b, according to the categorical distribution defined by the probability vector $\boldsymbol{\pi} = (\pi^j)_{j\in\mathcal{B}}$.

As explained in section 4.2.3, drawing according to a discrete distribution is made through a pair of (B, r) where B is a deterministic function and r some random values. A smart and intuitive way to do so is to divide an interval $[0, 1]$ into $|\mathcal{B}|$ buckets of size $\boldsymbol{\pi}$, and then returning the index of the bucket in which a random variable with uniform distribution on $[0, 1]$ falls. This operation, called Stair, is a function of $\boldsymbol{\pi}$ and the random variable u. The function is plotted for $\mathcal{B} = \{-1, 0, +1\}$ and for a given $\boldsymbol{\pi}$ on Figure 4.6. However, the derivative of Stair w.r.t. π_i^j is either equal to 0 or undefined, which makes the gradient irrelevant for gradient descent.

An alternative approach proposed in [70], is to draw $\mathbf{g} = (g^j)_{j\in\mathcal{B}}$ a vector of independant number sampled from the standard Gumbel distribution $G(0, 1)$, and to apply the following deterministic function:

$$b = \mathrm{HG}(\boldsymbol{\pi}, \mathbf{g}) = \arg\max_{j\in\mathcal{B}}(g^j + \log \pi^j). \tag{4.3.1}$$

In the above function HG (called *Hardmax Gumbel*), the arg max can be conveniently replaced by the softmax function:

$$\mathrm{softmax}\,(z_1, \ldots, z_n) = \frac{1}{\sum_{k=1}^{n} e^{z_k}}(e^{z_1}, \ldots, e^{z_n}),$$

which is a well known approximation of arg max, as can be seen from

$$\lim_{\tau\to 0} \mathrm{softmax}\left(\frac{z_1}{\tau}, \ldots, \frac{z_n}{\tau}\right) = (0, 0 \ldots, 0, 1, 0, \ldots, 0),$$

where the 1 is on $\arg\max_i z_i$ position and τ controlling the smoothness of the approximation is called *temperature*.

Replacing arg max in Equation (4.3.1) by a softmax approximation

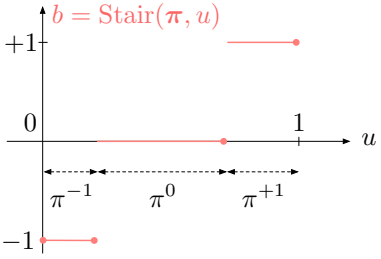


Figure 4.6: Plot of the Stair function with respect to its second variable u, for a given probability vector $\boldsymbol{\pi} = (\pi^j)_{j\in\{-1,0,1\}}$.

The Gumbel distribution with location μ and scale $\beta > 0$ is defined by its probability density function:

$$d(x; \mu, \beta) = \frac{1}{\beta} e^{-(z+e^{-z})}$$

with $z = \frac{x-\mu}{\beta}$. The probability density of the standard Gumbel distribution with $\mu = 0$ and $\beta = 1$ is plotted on Figure 4.7.

with temperature leads to :

$$\tilde{b}_\tau = \mathrm{SG}_\tau(\boldsymbol{\pi}, \mathbf{g}) = \sum_{j \in \mathcal{B}} j \; z^j, \tag{4.3.2}$$

$$\text{with } \mathbf{z} = \mathrm{softmax}\left(\frac{\mathbf{g} + \log \boldsymbol{\pi}}{\tau}\right). \tag{4.3.3}$$

The last Equation (4.3.2) is easily differentiable with respect to $\boldsymbol{\pi}$ since the calculation can treat the random numbers $\mathbf{g}$ as constants, but introduces a bias when $\tau > 0$. Figure 4.8 gives the comparison of the Hardmax and the Softmax Gumbel, where the last one is differentiable w.r.t. distributions parameters.

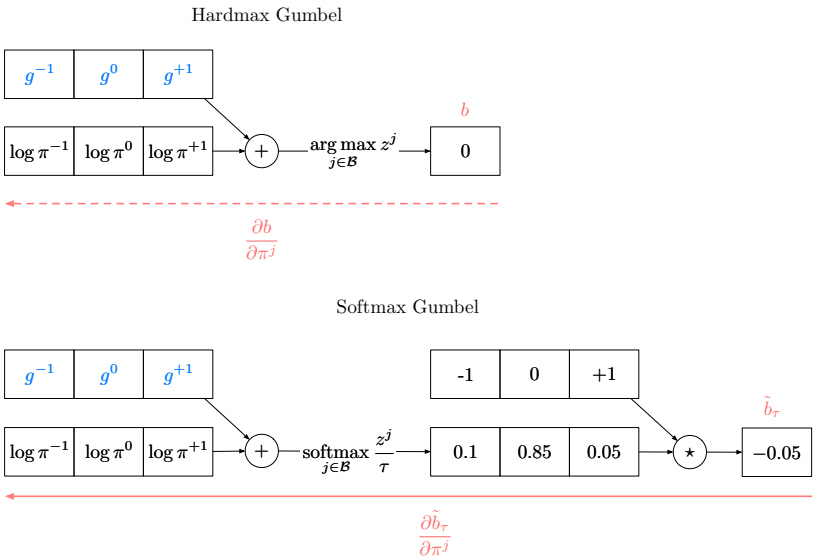


Figure 4.8: (Top) Pipeline to draw in category set $\mathcal{B} = \{-1, 0, +1\}$ according to probabilities $(\pi^j)_{j \in \mathcal{B}}$ with $(g^j)_{j \in \mathcal{B}}$ sampled from $G(0, 1)$. At the end, the $b \in \mathcal{B}$ modification is categorical but non-differentiable w.r.t. probabilities. (Bottom) Reparametrization trick with temperature τ, by replacing the non-differentiable $\arg\max$ by a softmax function. The final modification $\tilde{b}_\tau$ is continuous, but differentiable w.r.t. π^j. The $\cdot$ operation is a dot product. The backpropagation, illustrated by a plain colored arrow, is therefore possible.

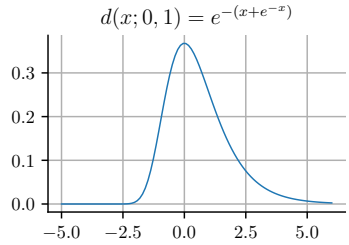


Figure 4.7: Probability density function of the standard Gumbel distribution.

Figure 4.9 offers a visualization of the influence of τ on the output of the Softmax-Gumbel (SG) function, for a fixed realization of a random vector $\mathbf{g}$ and fixed probability vector $\boldsymbol{\pi}$.

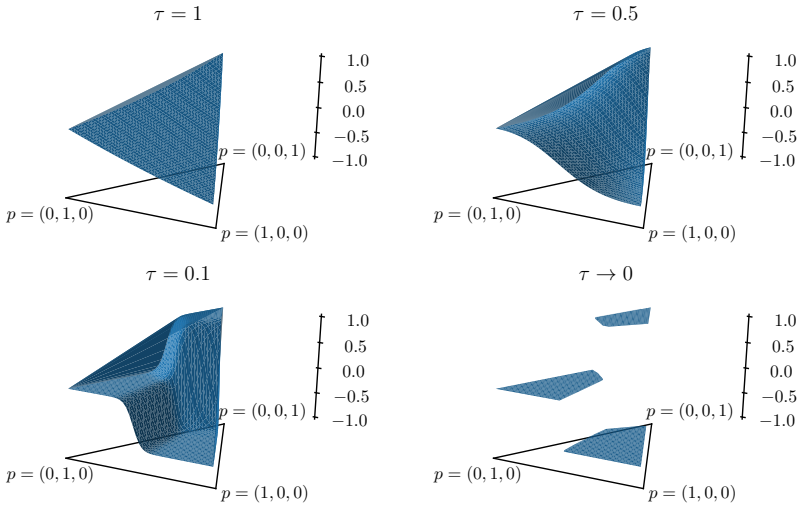


Figure 4.9: For a given value of triplet $g = (g^{-1}, g^0, g^{+1})$ (where $g^j \sim G(0, 1)$ are independently drawn from Gumbel standard distribution), value of the modification $\tilde{b}_\tau = \mathrm{SG}(p, g)$ is plotted the z-axis for all possible triplets of probabilities $p = (p^{-1}, p^0, p^{+1})$, and for 4 values of τ. The triplets are plotted in the trilinear coordinate system.

4.3.2 *Using the gradient of an implicit function to backpropagate through steps 2-1.*

Last step is about computing $\frac{d\boldsymbol{\pi}}{d\boldsymbol{\rho}}$. The chain rule gives, because $\pi_i^j = p_i^j(\boldsymbol{\rho}, \Lambda(\boldsymbol{\rho}, |\mathbf{m}|))$:

$$\begin{aligned} \frac{d\boldsymbol{\pi}}{d\boldsymbol{\rho}} &= \frac{\partial \boldsymbol{\pi}}{\partial \boldsymbol{\rho}}\frac{\partial \boldsymbol{\rho}}{\partial \boldsymbol{\rho}} + \frac{\partial \boldsymbol{\pi}}{\partial \lambda}\nabla_{\boldsymbol{\rho}}\lambda \\ &= \frac{\partial \boldsymbol{\pi}}{\partial \boldsymbol{\rho}} + \frac{\partial \boldsymbol{\pi}}{\partial \lambda}\nabla_{\boldsymbol{\rho}}\lambda. \end{aligned} \tag{4.3.4}$$

Although function Λ is implicit, its gradient $\nabla_{\boldsymbol{\rho}}\Lambda(\rho, |\mathbf{m}|))$ can be computed. Recall that for a given $\boldsymbol{\rho}$, λ is a solution of an entropy constraint (Equation (4.2.5)), therefore it holds that

$$H(P_{\mathbf{b}}(\boldsymbol{\rho}, \lambda)) = H(P_{\mathbf{b}}(\boldsymbol{\rho}, \Lambda(\rho, |\mathbf{m}|))) = |\mathbf{m}|,$$

and therefore $H(P_{\mathbf{b}}(\boldsymbol{\rho}, \Lambda(\boldsymbol{\rho}, |\mathbf{m}|))) - |\mathbf{m}| = 0$. Applying the chain rule to this equation gives :

$$\frac{d}{d\boldsymbol{\rho}}H(P_{\mathbf{b}}(\boldsymbol{\rho}, \Lambda(\rho, |\mathbf{m}|))) = \nabla_{\boldsymbol{\rho}}H(\boldsymbol{\pi})\frac{\partial \boldsymbol{\rho}}{\partial \boldsymbol{\rho}} + \frac{\partial H(\boldsymbol{\pi})}{\partial \lambda}\nabla_{\boldsymbol{\rho}}\lambda = 0,$$

from which the desired gradient of $\Lambda(\boldsymbol{\rho}, |\mathbf{m}|)$ can be expressed as

$$\nabla_{\boldsymbol{\rho}}\lambda = -\left(\frac{\partial H(\boldsymbol{\pi})}{\partial \lambda}\right)^{-1}\nabla_{\boldsymbol{\rho}}H(\boldsymbol{\pi}). \tag{4.3.5}$$

4.3.3 *Final approximation of the gradient, with continuous modifications*

In order to be able to compute $\frac{\partial \mathbf{b}}{\partial \boldsymbol{\pi}}$, we propose to use the Softmax gumbel approximation, such that discrete modifications are no longer needed but smooth ones $\tilde{\mathbf{b}}_\tau$ controlled by a temperature τ. The stego, obtained by the summation of the modifications to the cover, are therefore noted $\tilde{\mathbf{y}} = \mathbf{x} + \tilde{\mathbf{b}}_\tau$. We can also show that $\frac{\partial \tilde{\mathbf{y}}^k}{\partial \tilde{\mathbf{b}}_\tau^k}$ equals to the identity matrix, because $\frac{\partial (x_i + b_i)}{\partial b_j} = [i = j]$. So we can remove it from the chain rule formula.

Combining Equation (4.2.8) with Equations (4.2.9), (4.2.10), (4.3.4) and (4.3.5) yields to a closed form expression for the gradient :

$$\begin{aligned} &\nabla_{\rho} \mathop{\mathbb{E}}_{\tilde{\mathbf{b}}_\tau \sim P_{\tilde{\mathbf{b}}_\tau}(\tilde{\mathbf{b}}_\tau | \boldsymbol{\rho}, \lambda)} [f(\tilde{\mathbf{y}})] \approx \\ &\left(\frac{1}{K}\sum_{k=1}^{K} \nabla_{\tilde{\mathbf{y}}^k} f(\tilde{\mathbf{y}}^k)\frac{\partial \tilde{\mathbf{b}}_\tau^k}{\partial \boldsymbol{\pi}}\right)\left(\frac{\partial \boldsymbol{\pi}}{\partial \boldsymbol{\rho}} - \frac{\partial \boldsymbol{\pi}}{\partial \lambda}\left(\frac{\partial H(\boldsymbol{\pi})}{\partial \lambda}\right)^{-1}\nabla_{\boldsymbol{\rho}}H(\boldsymbol{\pi})\right). \end{aligned} \tag{4.3.6}$$

For practical computation, this formula is not required, as it can be handled automatically by auto-differentiation. Only Equation (4.3.5) is needed to specify the gradient of a non explicitly differentiable function Λ.

However, for the curiosity of the reader, we show below the explicit value, for a unique sample of stego, the value of $\frac{\partial f(\tilde{\mathbf{y}})}{\partial \rho_k^l}$, expressed from all computations on the following table.

$\frac{\partial \tilde{b}_{\tau,k}}{\partial \pi_i^j}$	$[k=i]\frac{1}{\tau}\frac{z_i^j}{\pi_i^j}(j-\tilde{b}_{\tau,i}) \qquad \text{where } z_i^j = \frac{e^{\frac{g_i^j+\log \pi_i^j}{\tau}}}{\sum_{l\in\mathcal{B}} e^{\frac{g_i^l+\log \pi_i^l}{\tau}}}$
$\frac{\partial \pi_i^j}{\partial \rho_k^l}$	$\lambda[k=i]\pi_i^j\left(\pi_i^l-[l=j]\right)$
$\frac{\partial \pi_i^j}{\partial \lambda}$	$\pi_i^j\left(\sum_{m\in\mathcal{B}}\rho_i^m\pi_i^m-\rho_i^j\right)$
$\frac{\partial H(\boldsymbol{\pi})}{\partial \lambda}$	$-\sum_{i=1}^{n}\sum_{j\in\mathcal{B}}\frac{\partial \pi_i^j}{\partial \lambda}\left(1+\log \pi_i^j\right)$
$\frac{\partial H(\boldsymbol{\pi})}{\partial \rho_k^l}$	$-\sum_{i=1}^{n}\sum_{j\in\mathcal{B}}\frac{\partial \pi_i^j}{\partial \rho_k^l}\left(1+\log \pi_i^j\right)=-\lambda\sum_{j\in\mathcal{B}}\pi_k^j\left(\pi_k^l-[l=j]\right)\left(1+\log \pi_k^j\right)$

Table 4.1: Analytic formulas of the gradients needed to compute $\nabla_{\boldsymbol{\rho}} f(\tilde{\mathbf{y}})$. Details of the calculations are shown in Appendix 5.

Finally, we can write:

$$\frac{\partial f(\tilde{\mathbf{y}})}{\partial \rho_k^l} = \frac{1}{\tau}\sum_{i,j}\frac{\partial f(\tilde{\mathbf{y}})}{\partial \tilde{y}_i}\frac{z_i^j}{\pi_i^j}\left(j-\tilde{b}_{\tau,i}\right)\left[\frac{\partial \pi_i^j}{\partial \rho_k^l}+\left(\frac{\partial H(\boldsymbol{\pi})}{\partial \lambda}\right)^{-1}\pi_i^j\left(\sum_n \rho_i^n\pi_i^n-\rho_i^j\right)\left(\sum_n \frac{\partial \pi_k^n}{\partial \rho_k^l}\left(1+\log \pi_k^n\right)\right)\right]. \tag{4.3.7}$$

4.3.4 *Optimizing embedding costs*

Results presented in the above sections allows us to efficiently approximate the gradient of Equation (4.2.3) by estimating a gradient of its smooth approximation while complying with a constraint on the entropy. It allows to use gradient descent method to minimise detectability with respect to all detectors in a set $\mathcal{F}^k$ as needed in min max protocol.

The proposed algorithm with pseudocode shown in Algorithm 4 uses a continuous approximation of discrete embedding changes to optimise the embedding costs ρ. In every iteration, it checks if the detectability of stego images with discrete embedding costs is below some threshold. If yes, the algorithm terminates; otherwise, it continues. If the detectability of stego images with a continuous approximation is below a given threshold, the temperature is halved. In practice, there is also a limit on the maximum number of iterations. All expectations in the pseudocode are estimated from a single sample, as described in section 4.2.3. The threshold on detectability is the detectability of the unmodified cover object.

The progress of the proposed algorithm on minimising the detectability of a single stego object against a single detector is shown in Figure 4.10. Although the optimisation uses continuous approximation of stego objects (blue line), the main goal is to create stego objects

Data: A JPEG cover image $\mathbf{x}$, initial embedding costs ρ^0, initial τ^0
Result: An adversarial embedding costs $\boldsymbol{\rho}$
$\boldsymbol{\rho} \leftarrow \boldsymbol{\rho}^0$;
$\tau \leftarrow \tau^0$;
$\tilde{o} \leftarrow \max_i \mathbb{E}_{\tilde{\mathbf{b}}_\tau \sim P(\lambda, \boldsymbol{\rho})}[f^i(\mathbf{x} + \tilde{\mathbf{b}}_\tau) - f^i(\mathbf{x})]$;
$o \leftarrow \max_i \mathbb{E}_{\mathbf{b} \sim P(\lambda, \boldsymbol{\rho})}[f^i(\mathbf{x} + \mathbf{b}) - f^i(\mathbf{x})]$;
while *True* **do**
  while $\tilde{o} > 0$ *and* $o > 0$ **do**
    Update $\boldsymbol{\rho}$ by one step of gradient descend with $\frac{\partial \tilde{o}}{\partial \boldsymbol{\rho}}$;
    $\tilde{o} \leftarrow \max_i \mathbb{E}_{\tilde{\mathbf{b}}_\tau \sim P(\lambda, \boldsymbol{\rho})}[f^i(\mathbf{x} + \tilde{\mathbf{b}}_\tau) - f^i(\mathbf{x})]$;
    $o \leftarrow \max_i \mathbb{E}_{\mathbf{b} \sim P(\lambda, \boldsymbol{\rho})}[f^i(\mathbf{x} + \mathbf{b}) - f^i(\mathbf{x})]$;
  end
  if $o \leq 0$ **then**
    Return $\boldsymbol{\rho}$
  else
    while $\tilde{o} \leq 0$ **do**
      $\tau \leftarrow \frac{\tau}{2}$;
      $\tilde{o} \leftarrow \max_i \mathbb{E}_{\tilde{\mathbf{b}}_\tau \sim P(\lambda, \boldsymbol{\rho})}[f^i(\mathbf{x} + \tilde{\mathbf{b}}_\tau) - f^i(\mathbf{x})]$;
    end
  end
end

Algorithm 4: The proposed algorithm optimizing embedding costs to minimize detectability of a stego object with respect to a set of steganalyzers $\mathcal{F}^k$.

with discrete embedding change (orange line). We can observe that in the very beginning, when temperature is high, there is a big difference between detectability of continuous approximations and that of real stego objects. But as the algorithm progresses and temperature decreases, this difference becomes negligible.

There are still hyperparameters in the Backpack algorithm:

- in practice, the *while* loops should be monitored by a maximum step, to ensure the algorithm will end. It is called N_{max}.
- the initial tempeature τ_0, decreasing policy for the temperature (ie when and how to decrease).
- the stopping condition: when to exit the gradient descent?
- the number of samples used to estimate the gradient.
- the learning rate of the optimizer, which achieves a step of the gradient descent.

The proposed algorithm is iterative; therefore, it does not suffer the weakness of ADV-EMB described in section 3.9, and it is well suited to minimise detectability measured as a maximum over a set of steganalyzers.

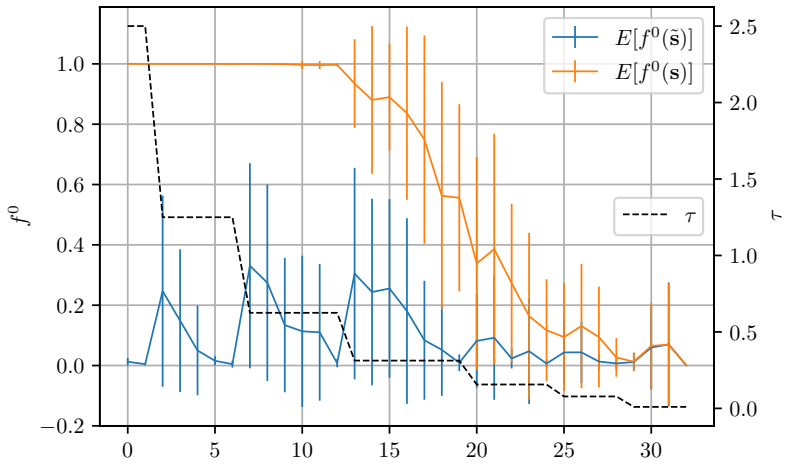


Figure 4.10: Effect of decreasing the temperature τ during optimization of a embedding costs for a given cover image. Is plotted on the left y-axis the average and variance of detectability (1 for stego class and 0 for cover class) given by classifier f^0 (for which $f^0(\mathbf{x}) = 0$) over 20 sampled simulated continuous stego object (blue plot where $\tilde{\mathbf{s}} = \mathbf{x} + \tilde{\mathbf{b}}_\tau$) and for 20 sampled simulated discrete stego object (orange plot, where $\mathbf{s} = \mathbf{x} + \mathbf{b}$), over the 33 steps of optimization on the x-axis.

4.4 Experimental evaluation

Backpack is below compared to ADV-EMB by each method implementing step 2 in the min max protocol (see section 3.2.4). Note that due to the weakness described in section 3.9 the ADV-EMB algorithm is computationally less expensive, since it is sufficient to optimise it against the last steganalyzer f^k, whereas Backpack needs to be optimised with respect to all classifiers $f \in \mathcal{F}^k$.[2]

[2] Note that due to the max function, it attacks in each iteration of Algorithm 4 a single steganalyzer from the set $\mathcal{F}^k$, but this single classifier is potentially different at every iteration.

4.4.1 *Experimental settings*

Images The experiments use the JPEG version of the BossBase database [71] of size 512×512 in grayscale format and compressed with Quality Factor (QF) 100 and 75. All images are embedded using an embedding rate of 0.4 bits per non-zero AC DCT coefficient (bpnzAC) at each algorithm iteration.

Steganalysis A proper evaluation of min max protocol (and its variants) requires two sets of steganalyzers and their overlap depends on who knows what. The first set of classifiers, $\mathcal{F}$ is available to Alice, who runs the min max protocol. In this work, this set contains all classifiers with XuNet architecture [41] (differing in weights). The second set, $\tilde{\mathcal{F}}$, of classifiers is available to Eve. In our experiments, $\tilde{\mathcal{F}}$ contains all classifiers with SrNet and XuNet architectures, classifiers trained with DCTR [33] or GFR [34] features. XuNet and SrNet were implemented in TensorFlow [63]. This experimental setup allows investigating two different setups, which practically express the assumptions that whether or not Alice knows which class of steganalyzers Eve uses.

At each iteration of the min max protocol, a new steganalyzer f^k is trained by classifying cover objects $\mathcal{C}$ and stego objects $\mathcal{S}^k$ created in the previous iteration at the second step of the min max protocol. Steganalyzers are trained on full-size images of 512×512 coefficients, 2×4000 cover and stego objects for training, 2×1000 for validation

set and using remaining 2×5000 to estimate error rates. The training database is shuffled after each epoch. In each batch, we apply data augmentation based on random mirroring and rotation of the batch images by 90 degrees. 280 epochs are used for training using Adam optimizer [72]. The configuration achieving the best validation accuracy is used as the result of training. XuNet, the classifier is trained starting with randomly initialised weights (zero-mean Gaussian with standard deviation 0.01), the initial learning rate is set to 0.001 and decreased after each 5000 steps to 0.9 times the current value. The remaining parameters of Adam are kept to default setting. The size of mini-batch is 32 (16 cover-stego pairs). The configuration of SrNet is the one proposed in the paper [43], except the training uses 280 epochs. The size of mini-batch is 16 (8 cover-stego pairs).

Optimisation of embedding costs Both compared methods requires initialisation of embedding costs, for which those of J-Uniward [73] were used (this has been done in [31]). The ADV-EMB method for adjusting costs is implemented as described in section 1.5.10. Backpack uses Adam [72] with a learning rate of 0.05 to optimize the embedding costs ρ in Algorithm 4. The maximum number of steps during Backpack optimisation was 500 until iteration 5, and was 2000 until the end of the protocol. Gradients of expected error (Equation (4.2.9)) are computed with $k = 30$ samples until fourth iteration of min max protocol, with $k = 20$ samples until its eight iteration, and then with $k = 10$ samples. Although a single sample is frequently sufficient, more samples improve predicted gradients' accuracy and can be calculated in parallel on the GPU in the same batch. However, as min max protocol progresses, the gradient needs to be calculated with increasingly more models, which occupies the memory of GPU and therefore, we had to decrease the number of samples progressively. The initial temperature was set to $\tau^0 = 10$. The error of steganalyzers was measured by Equation (3.5.1), which is the usual average error on the cover and stego objects assuming the equal prior probability of their occurrence (denoted as P_{err}). Since the goal of steganography is to be undetectable, a higher value is better.

4.4.2 *Discussion of results*

QF	h_{emb}	P_{err} (%)			
		XuNet	SrNet	DCTR	GFR
100	J-Uniward ($k = 0$)	16.9	13.1	26.6	26.8
	ADV-EMB ($k = 8$)	26.5	18.9	26.5	32.4
	Backpack ($k = 8$)	**37.4**	**25.3**	**30.7**	**37.3**
75	J-Uniward ($k = 0$)	7.5	6.0	16.2	10.0
	ADV-EMB ($k = 7$)	22.0	10.7	26.7	25.8
	Backpack ($k = 7$)	**47.6**	**15.6**	**32.9**	**31.5**

Table 4.2: Values of P_{err} plotted in Figure 4.11 at $k = 0$ and $k = 7$ or $k = 8$ for QF 75 or QF 100, for both ADV-EMB and Backpack.

The main bulk of experimental results are presented in Figure 4.11

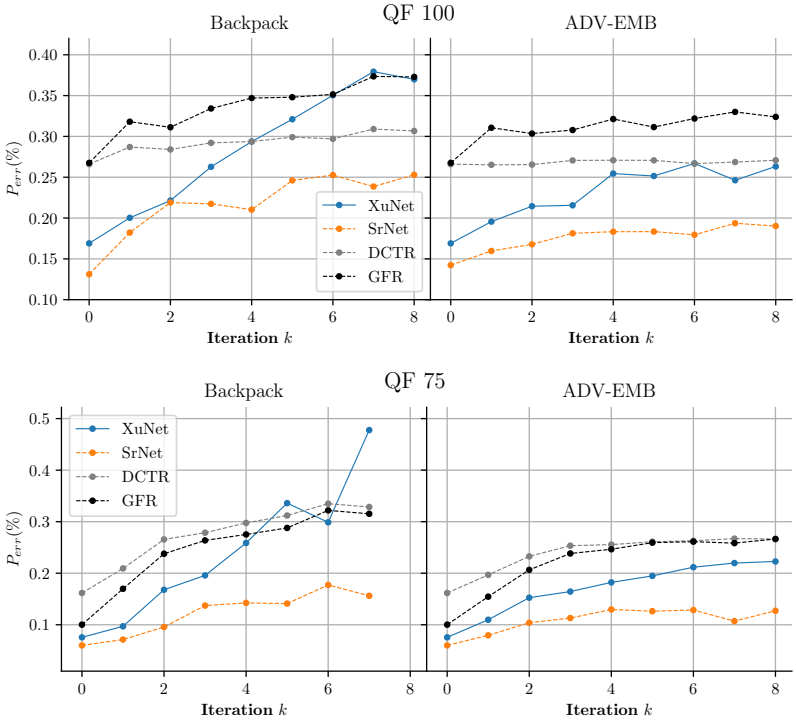


Figure 4.11: P_{err} of test sets w.r.t iterations of the protocol with QF 100 (top line) or QF 75 (bottom line), an embedding rate of 0.4 bpnzAC, cost initialized with J-Uniward and applied with our attack (left column) or ADV-EMB attack (right column). Assumed class of detectors is XuNet architecture, and real detectors are XuNet, SrNet, DCTR and GFR.

and in Table 4.2 showing error P_{err} of XuNet, SrNet, DCTR, and GFR steganalyzers on testing data with respect to the iteration of min max protocol. The proposed Backpack method is superior to the ADV-EMB. The P_{err} error of a XuNet steganalyzer trained after eight iterations is 37% on images created with the proposed method while it is 26% on those created by ADV-EMB (also after eight iterations) and 16% on those created by J-Uniward. This means that if [2] is considered by state of the art, the proposed method has improved by 11% (as measured by XuNet for which it has been optimised). These results are for JPEG 100 and in the optimistic case for Alice where she knows which type of steganalyzer Eve will use (but Eve optimises her steganalyzer on Alice's stego images from her final iteration such that Kerckhoffs' principle is not violated). It is interesting to observe that even though Alice is not explicitly optimising against SrNet, DCTR, and GFR steganalyzers, she is still improving the security of her steganographic algorithm with respect to them, although the curve is not as steep as that for XuNet.

The evolution of P_{err} also suggests that GFR steganalyzers relies on a similar type of information as XuNet (on JPEG images with QF 100), but DCTR and SrNet use different types, as the improvement in the security is not as high.

Experimental results on JPEG images with QF 75 are similar to those on QF 100 though there is a notable difference in the behaviour of steganalyzer using DCTR features. On QF 100, this steganalyzer is almost insensitive to the improvement in security with respect to XuNet, whereas on QF 75, it reflects the improvement. It suggests

that at QF 100, it is detecting artefacts, which are not present at QF 75, and XuNet cannot see these artefacts. It might be caused by rounding artefacts described in [74].

Unlike ADV-EMB, Backpack does not contain any regularisation minimising the number of modified coefficients. For example, it can be trivially added by defining a prior on the distribution of embedding changes, but we believe it should be learned from data rather than added explicitly. Moreover, the min max protocol should theoretically correct too detectable embedding changes in subsequent iterations. Nevertheless, the steady increase in steganographic security as measured by XuNet steganalyzers do not indicate that such regularisation is needed.

4.5 Additional analysis of the results

4.5.1 *Performance of Backpack compared to ADV-EMB*

We can observe that Backpack solves a major weakness of ADV-EMB highlighted in section 3.9. Figure 4.12, compared to Figure 3.22, shows better performance at attacking all classifiers in the set of Eve's available actions.

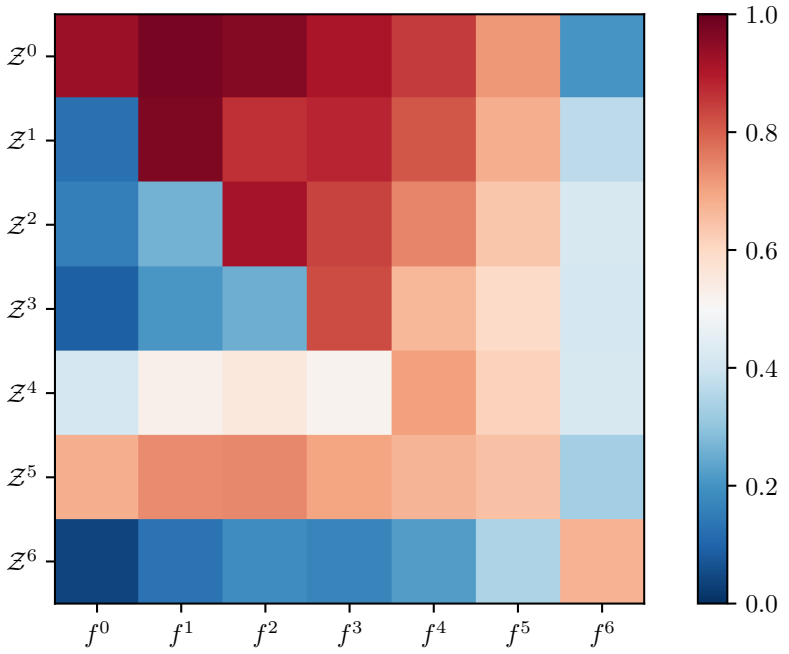


Figure 4.12: For the protocol with Backpack with JPEG images at QF 75 and payload 0.4 bpnzAC, with J-Uniward and XU-Net, average detectability given by each classifier f^j (columns) evaluated on each adversarial stego database $\mathcal{Z}^i$ (rows). Blue color is for images detected as cover (the probability of stego class is below 0.5), whereas red if for images classified as stego.

Indeed, and contrary to ADV-EMB, until iteration 4 the images in the set $\mathcal{Z}^k$ defeat on average all previous classsifier $f^l, l \leq k$ (see the blue cells). With the same condition of experiment, the task is more and more difficult, it is why $\mathcal{Z}^5$ do not provide as good results as for lower iteration. At iteration 6, we changed the parameters of Backpack and increased the number of steps for the gradient descent. It gives a more powerful attack, as you can see the last row of the image.

4.5.2 *Analysis of the correlations between DCT coefficients*

Among the several steganographic techniques, we saw in section 1.5.7 that synchronising the modifications while embedding can increase the security of the scheme, for example by using lattices. We show in this section that it can also be achieved via the definition of an *asymmetric* additive distortion function, and it is why we observe correlations for the embedding with Backpack.

The correlation between two random variables X and Y with expected values μ_X and μ_Y and standard deviations σ_X and σ_X is defined as:

$$\text{corr}(X,Y) = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\text{E}\left[(X-\mu_X)(Y-\mu_Y)\right]}{\sigma_X \sigma_Y}. \tag{4.5.1}$$

We can show that symmetric costs can not introduce correlations between coefficients because the covariance between two modes is equal to 0[3]. It is the case of J-Uniward (see section 1.5.4) with ternary embedding, because it gives a *symmetric* additive distortion function, i.e. $\rho_i^{-1} = \rho_i^{+1}$ for every coefficient i.

ADV-EMB (see section 1.5.10) might favour correlations for two reasons. First, because the message is embedded sequentially in two steps like it is done in synchronisation with lattices: (i) first piece embedded in the common group, then (ii) last piece embedded in the adjustable group, where the costs have been modified according to modifications made in the common group. Second, because the costs are asymmetric, like for Backpack.

Obviously, ADV-EMB introduces asymmetric costs, i.e. costs $\rho^{-1} \neq \rho^{+1}$ because of its update rule shown in Equations (3.1.1) and (3.1.2). The asymmetry on cost leads to an asymmetry on probabilities, which is plotted in Figure 4.13. It shows the log histograms of the differences between probabilities of $+1$ and -1 modifications of coefficient of several covers. A null difference for a coefficient is equivalent to symmetric costs. For both ADV-EMB and Backpack, there is a high quantity of differences close to 0, as a lot of costs are set to a high value in both directions. But the repartition of the differences is significantly different. In the case of ADV-EMB, we can observe that the absolute difference cannot be higher than 0.28. It might be due to the update rule of ADV-EMB, which makes the ratio between ρ^{-1} and ρ^{+1} equals to either 1, $1/\alpha^2$ or α^2. In the case of Backpack, the gradient descent may lead to considerable differences between probabilities, as we can see some reaching 0.5 in absolute value.

We can also observe that the quantity of asymmetric costs increases with iteration k, i.e. there are more coefficients with asymmetric costs in $\mathcal{Y}^1$ than in $\mathcal{Y}^0$, and more in $\mathcal{Y}^2$ than in $\mathcal{Y}^1$, etc. For ADV-EMB, this might be due to the increase of the number of costs modified with iterations k, as it was shown in Figure 3.19. For Backpack, this might be because more and more steps of the gradient descents might be required.

By analysing the covariance matrix of the stego signal of quantised

[3] Let us consider ternary embedding, and B_i the random variable modelling the modification made to a mode i. Let's note $b_i^{(k)}$ the random variable modeling the modification made to a specific coefficient k belonging to mode i (there are N samples for each mode i). We have

$$\text{E}[b_i^{(k)}] = \pi_i^{+1,(k)} - \pi_i^{-1,(k)} = \mu_i^{(k)},$$

so

$$\text{E}[B_i] = \frac{1}{N}\sum_k \mu_i^{(k)} = \mu_i.$$

We have also

$$\begin{aligned}\text{E}[b_i^{(k)} b_j^{(k)}] &= \pi_i^{-1,(k)}\pi_j^{-1,(k)} + \pi_i^{+1,(k)}\pi_j^{+1,(k)} \\ &\quad - \pi_i^{-1,(k)}\pi_j^{+1,(k)} - \pi_i^{+1,(k)}\pi_j^{-1,(k)} \\ &= \mu_i^{(k)}\mu_j^{(k)},\end{aligned}$$

.

then

$$\begin{aligned}\text{cov}(B_i,B_j) &= \text{E}[B_iB_j] - \text{E}[B_i]\text{E}[B_j] \\ &= \frac{1}{N}\sum_{k=1}^{N}\mu_i^{(k)}\mu_j^{(k)} - \mu_i\mu_j\end{aligned}$$

.

For symmetric costs, $\mu_i^{(k)} = 0 \forall i$, so $\text{corr}(B_i,B_j) = \text{cov}(B_i,B_j) = 0$.

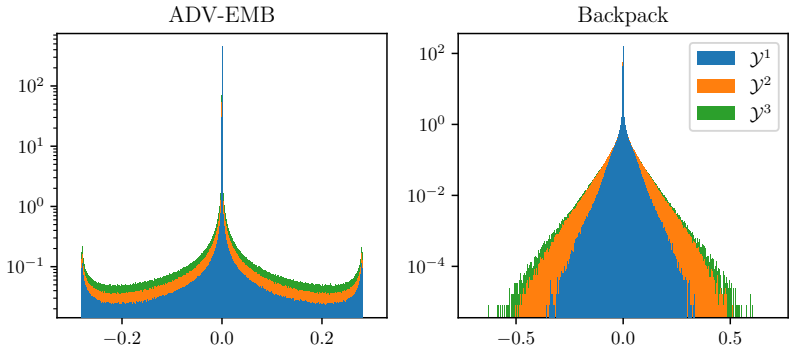


Figure 4.13: Log histograms of $\{\pi_i^{-1} - \pi_i^1\}$ for DCT coefficients of 100 cover images, obtained at iteration 1, 2 and 3 for protocol with (left) ADV-EMB or (right) backpack, for images at QF 75 with payload 0.4 bpnzAC.

JPEG coefficients (i.e. $\mathbf{b} = \mathbf{y} - \mathbf{x}$ the signal added to the JPEG Cover image to create the Stego image), we highlight the fact that the stego signal exhibits correlations between coefficients. These weak correlations are within the same block (intra-block correlations) or between adjacent blocks (inter-block correlations). It is shown on Figure 4.14 how the correlations are plotted on a 256×256 grid.

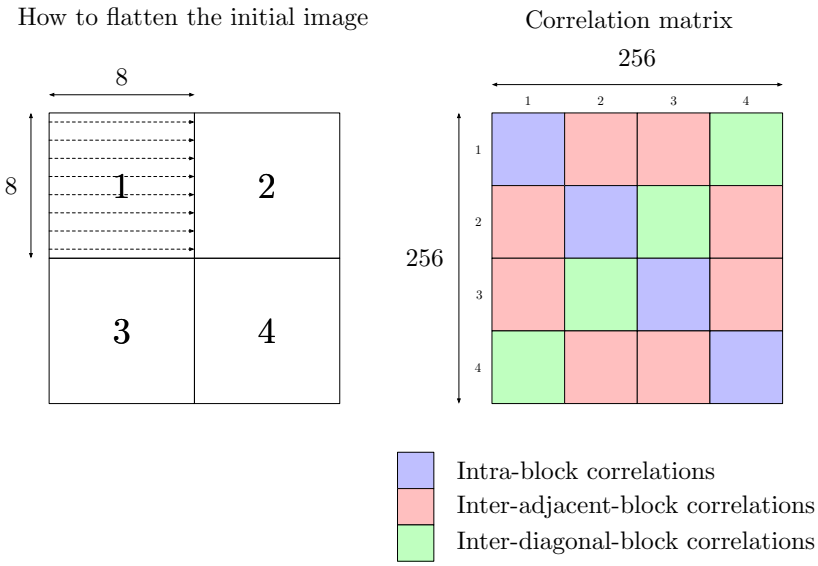


Figure 4.14: How the image is flatten in order to build the correlation matrix (left). If i is the index of the row or the column of a coefficient in the correlation matrix (right), $\lfloor i/64 \rfloor$, $\lfloor i/8 \rfloor$ and $i \mod [8]$ give respectively the index of the block (1,2,3 or 4), the index of the row and the index of the column of the coefficient within its block, in the original image.

There is, as expected, no correlations for J-Uniward, but there are for ADV-EMB and Backpack. The correlation patterns are similar to the patterns of correlations analysed on the sensor noise in the DCT domain (see [75]). However, if in [75] these correlations have been shown to favour *continuities* between blocks, the correlations induced by adversarial embedding are on the opposite sign, and they code *discontinuities* between blocks.

An experiment consisting of generating first a cover image using J-Uniward and then computing an adversarial signal using projected gradient descent (PGD [76]) exhibits very similar patterns than with adversarial embedding (see figure 4.15). We consequently presume that the adversarial signal is generated in order to compensate for the

blocking artefacts created by the embedding. It would explain the sign of the correlations. Our rationale is that the proportion β of the DCT coefficients used to produce the adverse stego signal in ADV-EMB conveys the signal whose role is to remove the blocking artefacts generated by additive schemes such as J-Uniward.

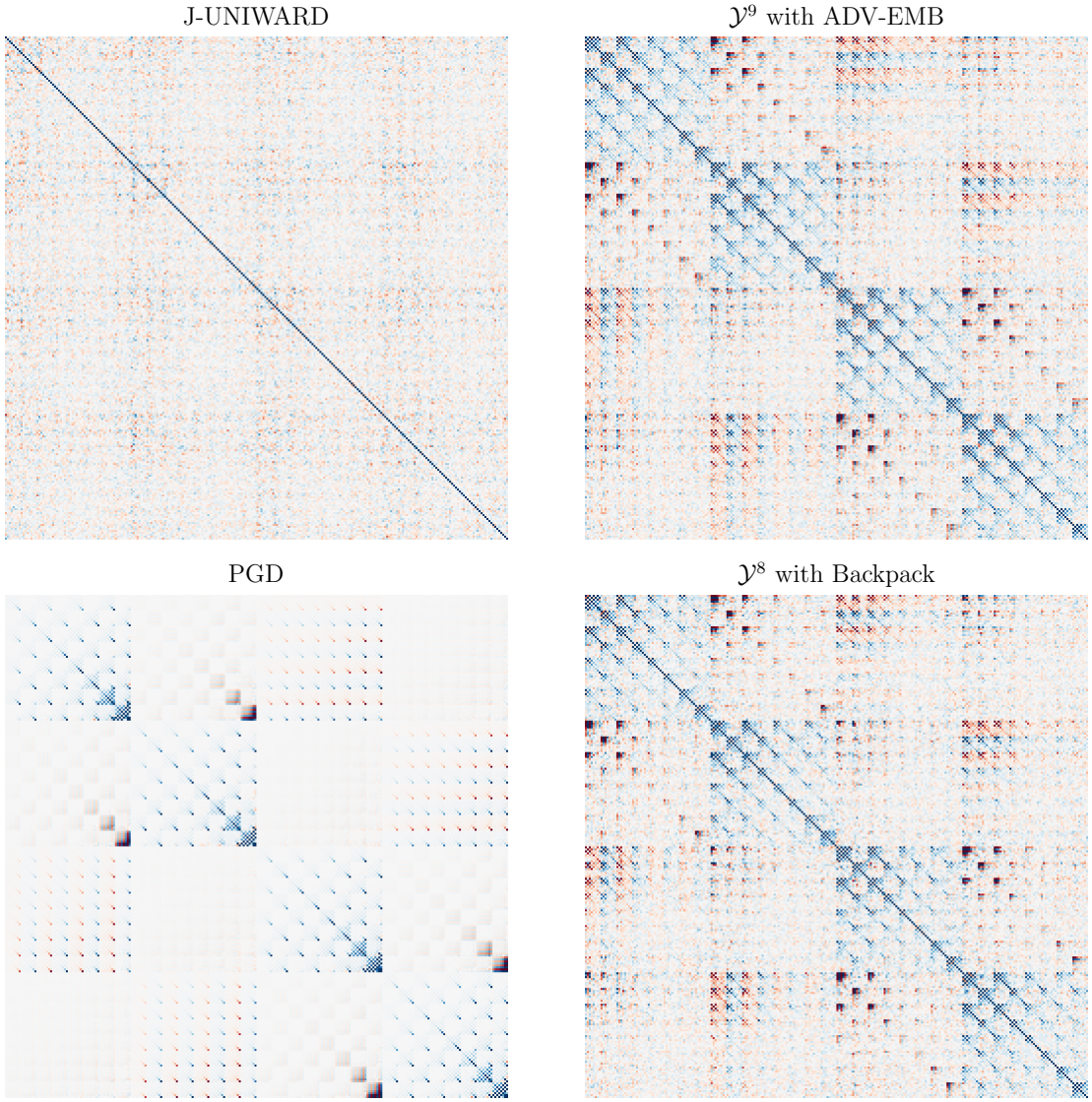


Figure 4.15: Covariance matrix computed by the stego signal $\mathbf{b} = \mathbf{y} - \mathbf{x}$, for (up-left) stegos obtained with J-Uniward, or (up-right) stegos in $\mathcal{Y}^9$ produced in the protocol with ADV-EMB, or (bottom-left) vector obtained with simple gradient descent from J-Uniward stegos or (bottom-right) stegos obtained with Backpack. Stegos at QF 100 paylaod 0.4 bpnzAC, using 7000 images of BOSSBase decomposed into non overlapping 16×16 blocks. (2 thresholding are applied to reduce the

4.5.3 *On the necessity to train correctly.*

Another experiment we conduct is optimizing the embedding to more architectures than XU-Net (like we did in chapter 3 with Figure 3.15). We add SRNet and Efficient-Net in the protocol and observe the evolution of the error rate in Figure 4.16 (left).

Experimental settings This implementation used Pytorch. Efficient-Net B0 (with a the first stride set to 1 to avoid the destruction of the

stego noise) was initialised with the training on ImageNet. Starting from iteration 4, the GPU can not load all models ($4 * 3 = 12$) to compute $\mathcal{Z}^4$. Therefore, we only defeat the last models from 3 past iterations (the 9 last models). At iteration 1, we estimated the gradient with $K = 5$ samples, at iteration 2, we used $K = 2$ examples, and starting from iteration 3, we used $K = 1$.

Computational cost At iteration 3, the optimisation of the cost map takes, in average, 16.12 minutes per image, on GPU Nvidia V100 with 16Go of memory.

Results For this experiment, because Efficient-Net might have difficulties converging, we fed the network with newly sampled stegos in each batch from the optimised cost maps obtained with Backpack. We can therefore apply Curriculum Learning (see section 2.4.1) easily because we simulate the embedding of a message of any length with the cost map. However, we observe a suboptimality associated with this training strategy. When we finetune the training by finally training on the stego images, we obtain a lower error rate (Figure 4.16, right).

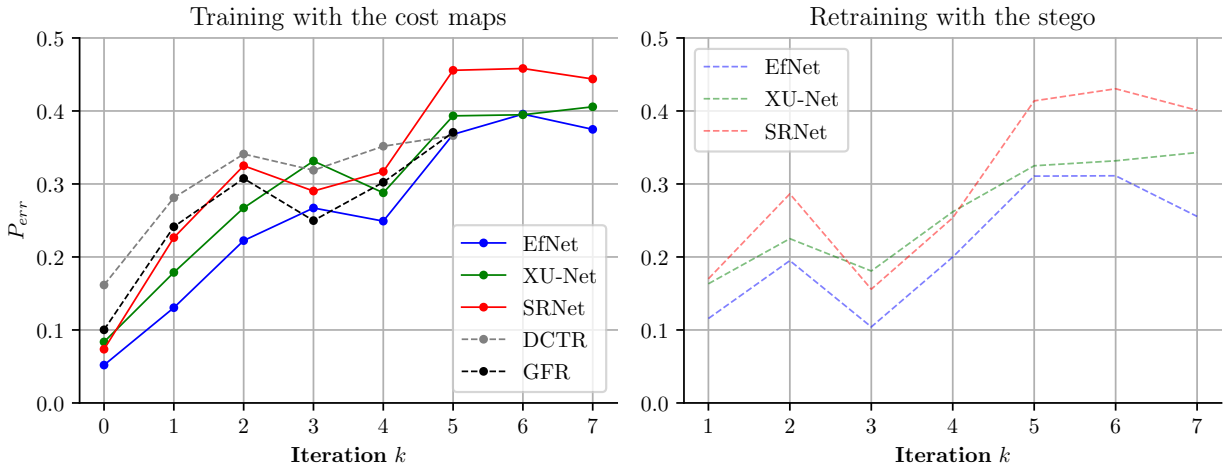


Figure 4.16: Experiment with $\mathcal{A}_e = \{\text{XU-Net}, \text{SRNet}, \text{EfNet}\}$ (the targeted classifiers, plain lines) and $\tilde{\mathcal{A}}_e = \mathcal{A}_e \cup \{\text{DCTR}, \text{GFR}\}$ (the untargeted classifiers, dashed lines), for images at QF 75, payload 0.4 bpnzAC. (Left) Error rate of classifiers when they are trained with newly sampled stegos at 0.4 bpnzAC from the optimized cost maps. (Right) Error rate of finetuned classifiers on the stegos obtained from the optimization.

First, it might be surprising because the stego images were initialy sampled from the cost maps. But the gradient descent Backpack, where an average estimates the expectation between K samples, could select particular stegos which are far from the probability distribution given by the optimised cost map. Secondly, it shows the importance of correctly training a classifier during the protocol, as the error rate can be overestimated.

We use the optimised cost map obtained in this experiment to evaluate a possible *payload mismatch*.

4.5.4 *Payload mismatch*

From the optimised cost map with respect to three architectures of classifiers obtained with Backpack, we investigate the following ques-

tion: “Are the resulted costs optimised to other payloads?” because we recall that we can use a cost map to embed a message of any length.

We proceed to the following experiment. We simulated embedding at payloads 0.1, 0.2, 0.3 and 0.5 in the cover from the optimized cost map obtained in experiment in section 4.5.3 at payload 0.4 bpnzAC. Then we trained the three architectures to detect the obtained stegos. We plot on Figure 4.17 the error rate of every model at each payload, compared to J-Uniward.

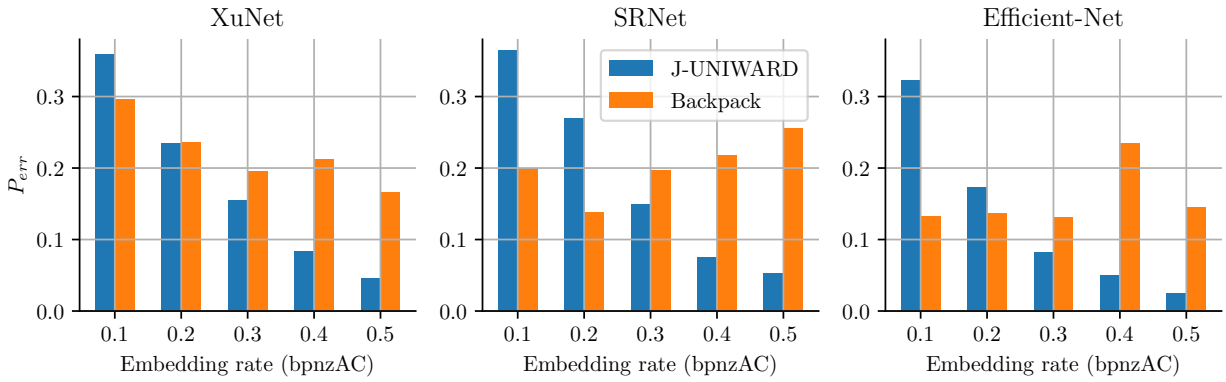


Figure 4.17: Error rate of three models (XuNet, SrNet and Efficient-Net) trained to detect stegos sampled for the cost maps at different embedding rates (from 0.1 to 0.5) where the cost are optimized with Backpack during a min max protocol at payload 0.4bpnzAC.

It is interesting to notice that J-Uniward has an intuitive behaviour, meaning that the smaller is the payload, the less detectable it is for each model. We do not observe the same behaviour for the optimised costs with Backpack. The Backpack costs give a worst result that J-Uniward for embedding rates of 0.1 and 0.2. For embedding rate starting from 0.3 it gives better results. For Efficient-Net, it is surprising how the cost seems overfitted to the embedding rate 0.4, because it is where it is the least undetectable.

4.5.5 *Where to stop?*

There is still a remaining question: how far should we cross the decision boundary when we fool a classifier?

This question is asked because we are solving a subgame of the real game with an infinite set of actions. When Alice plays the min max strategy, Eve can answer by creating a new action. There Alice hopes that the stego she creates will not be detected by the next detector. Eve’s utility function is the output probability of stego class of a detector. When the probability is below 0.5, the stego fools the classifier.

For our first contribution, we proposed to stop whenever the decision boundary is crossed, so whenever the stego probability class is below 0.5. We have the intuition that crossing the boundary too far would make the stego too detectable afterwards. For our second contribution, we decided to stop when the detectability of the stego becomes below the detectability of cover. It has the advantage of not modifying a stego whose cover is already misclassified. However, all these ideas are not relying on an optimised process, and it would be interesting to

evaluate the best boundary.

4.6 Conclusion and overview

This contribution framed *adversarial attacks* against steganalyzers into a perspective of the general steganographic problem, which is the minimisation of non-additive distortion functions. It has shown that adversarial attacks can be seen as an optimisation of an approximation of non-additive distortion function by its additive counterpart defined implicitly by costs of changing embedding coefficients.

The proposed method, called Backpack, relies on the fact that most state of the art steganalyzers, mainly those implemented by convolution neural networks, allow calculating gradients of their output with respect to the input (using back-propagation). Backpack approximates discrete embedding changes by samples from Gumbel-Softmax distribution, which is a standard approach in machine learning. It also uses differentiation of implicit function to handle constraints on message length effectively.

The experimental experimentally confirms the theoretical correctness of the approach. The security of a steganographic scheme as measured by XuNet on 512×512 JPEG images compressed with quality factor 100 with a payload of 0.4 bits per non-zero AC coefficient has increased to 37.3% whereas that of the previous state of the art known to authors was 26.5% under the same setting. Interestingly, although the steganographic algorithm was optimised with respect to XuNet steganalyzer, the security with respect to other steganalyzers achieved by SrNet, GFR or DCTR features has increased as well.

Conclusion and perspectives 5

At the beginning of this manuscript, we started by presenting naive steganography by hiding a bit of message per image coefficient, and we saw how highly detectable it is. We learned that it is crucial to adapt the embedding to the image content to respect its natural statistics. Distortion functions hold the role of deciding where to hide the data without being detectable.

On the other hand, convolutional neural networks perform well to spot a modified image. Given their learning examples, they can be trained to detect a specific embedding method. However, deep learning tools are susceptible to be targeted by an adversarial method, i.e. the stenographer can adapt its distortion function to avoid a specific classifier.

Each character can adapt its strategy depending on the action of the other. For this reason, we brought game theory notions to tackle this problem. We constructed a game for each cover image, where Alice can choose among multiple stego versions of the cover and where Eve can choose a detector. Eve's utility of the game is the probability of the stego class of the detector evaluated on the stego, and Alice's utility is the opposite. We, therefore, construct multiple zero-sum games.

In practice, the set of actions available to each player is enormous, so those games are not constructible. We propose instead to construct iteratively sub-games, where at each step, each player creates new action which is optimal to the game. Alice can create a new embedding function following a min max strategy via her adversarial scheme. Then Eve can learn a new classifier to detect the optimal stegos.

In that way, this thesis proposes an automatic procedure to improve the distortion function to fool a specific model. It is a very general scheme, and we can adapt it as long as Alice possesses an adversarial embedding scheme compatible with Eve's targeted detector.

In Chapter 3, we use for our experiments the ADV-EMB adversarial scheme, which can avoid a specific differentiable trained model. We choose XU-Net and SRNet for the detectors. In all our experiments, we show the performance of our protocol compared to the initial distortion J-UNIWARD or UERD. Specifically, the most secure version increases the error rate of SRNet for messages with payload 0.4 hidden in JPEGs by 10% comparing to J-UNIWARD: it jumps from 6.3% to 16.3%. We also show that the error rate of non-targeted classifiers (DCTR and GFR) increases.

In Chapter 4, we propose to improve the ADV-EMB algorithm by removing the heuristic. We propose Backpack, a method to differentiate through a probability distribution parameters. It enables

to optimize the expectation of the optimal detector with respect to the distortion function via gradient descent, while complaining to a constraint on entropy. In our experiments, we compare this new adversarial scheme in our protocol to ADV-EMB. We observe that Backpack increases the security and the error rate of classifiers no longer saturates with the iterations. For a protocol with images at QF 75, a payload of 0.4 against XU-Net, we reach an error rate of 47.6%.

Perspectives

An interesting perspective, introduced in Section 3.2.3, is the consideration of mixed strategy, on the opposite of pure strategies that we were playing for our two contributions. It is easily solvable, as the `scipy.linprog` library can find the Nash equilibrium because any two-players zero-sum game can be expressed as a linear program [52].

Another is motivated by the computational cost of the protocol. The main limitation is the computational cost of our technique (see Section 4.5.3). This is all the more serious as neither cost maps nor the classifiers obtained at the end of a protocol can be used to embed in another cover source or embed a message of another length.

First, it requires many GPU resources to backpropagate through models with many parameters for each image several times. Second, we show that the cost maps are not transferable to hide a message with other payloads. We show that the classifiers are no relevant adversaries to optimize the cost map for other cover sources. Therefore, it is required to re-run an entire protocol to adapt the cost map for new parameters (for example, changing the QF or the development process of the image), which is a costly operation. Because the protocol is specific to a payload and a cover source, it is very motivating to accelerate the convergence process.

Optimization of hyperparameters of Backpack Optimizing the adversarial embedding scheme (Alice's new action in the protocol) to produce stego not too detectable at the next step is a way to speed up the convergence. As explained in Section 4.3.4, there are hyperparameters in the Backpack algorithm: the maximum number of steps, the learning rate, the number of samples, the decreasing policy for the temperature and the stopping condition.

We suggested in 4.5.5 to optimize the stopping condition of the resolution of the min max. The convergence of the protocol might be improved if the resolution is optimized, such as images are not too detectable at the next step.

The Backpack technique requires to load mutliple models at the same time in the GPU, such as we can optimize the cost map with respect to the optimal detector. At some point, especially if we want to optimize w.r.t. different architectures, it is motivating to decrease the GPU loading. A solution could be to use *distillation*, which is the

process of transferring knowledge from a large model to a smaller one. The challenge it that smaller classifier should remain an interesting opponent, meaning that computing adversarial content to the distilled network should stay adversarial to the original networks.

Bibliography

[1] Solène Bernard, Patrick Bas, John Klein, et al. *Explicit optimization of min max steganographic game.* In: IEEE Transactions on Information Forensics and Security (2020).

[2] Solène Bernard, Tomás Pevny, Patrick Bas, et al. *Exploiting adversarial embeddings for better steganography.* In: Proceedings of the ACM Workshop on Information Hiding and Multimedia Security. 2019.

[3] Solène Bernard, Tomas Pevny, Patrick Bas, et al. *Utilisation d'insertions adverses pour améliorer la stéganographie.* In: GRETSI. Lille, France, 2019.

[4] Solène Bernard, Patrick Bas, Tomáš Pevny, et al. *Optimizing Additive Approximations of Non-Additive Distortion Functions.* In: Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security. 2021.

[5] Gustavus J Simmons. *The prisoners' problem and the subliminal channel.* In: Advances in Cryptology. Springer. 1984.

[6] Andrew Stockman and Lindsay T Sharpe. *The spectral sensitivities of the middle-and long-wavelength-sensitive cones derived from measurements in observers of known genotype.* In: Vision research (2000).

[7] Christian Mauer and Dietmar Wueller. *Measuring the spectral response with a set of interference filters.* In: Digital Photography V. International Society for Optics and Photonics. 2009.

[8] Thomas Young. *On the theory of light and colours.* In: (1802).

[9] James Clerk Maxwell. *XVIII.— Experiments on Colour, as perceived by the Eye, with Remarks on Colour-Blindness.* In: Transactions of the Royal Society of Edinburgh (1855).

[10] Michel Loève. *Elementary Probability Theory*. In: Probability Theory I. New York, NY: Springer New York, 1977. ISBN: 978-1-4684-9464-8.

[11] Joseph Fourier. *Théorie analytique de la chaleur.* Chez Firmin Didot, père et fils, 1822.

[12] Benjamin Arazi. *Two-dimensional digital processing of one-dimensional signal.* In: IEEE Transactions on Acoustics, Speech, and Signal Processing (1974).

[13] Nasir Ahmed, T. Natarajan, and K.R. Rao. *Discrete Cosine Transform.* In: IEEE Transactions on Computers (1974).

[14] Jessica Fridrich. *Steganography in digital media: principles, algorithms, and applications.* Cambridge University Press, 2009.

[15] Richard W Hamming. *Error detecting and error correcting codes.* In: The Bell system technical journal (1950).

[16] Claude Shannon. *A mathematical theory of communication.* In: Bell System Tech. J. (1948).

[17] Christian Cachin. *An information-Theoretic Model for Steganography.* In: Information Hiding: Second International Workshop IHW'98. Portland, Oregon, USA, 1998.

[18] Auguste Kerckhoffs. *La Cryptographie Militaire.* In: Journal des Sciences Militaires. 9. 1883.

[19] Tomáš Filler and Jessica Fridrich. *Gibbs construction in steganography.* In: IEEE Transactions on Information Forensics and Security (2010).

[20] Tomáš Filler, Jan Judas, and Jessica Fridrich. *Minimizing additive distortion in steganography using syndrome-trellis codes.* In: Information Forensics and Security, IEEE Transactions on (2011).

[21] Tomáš Pevny, Tomáš Filler, and Patrick Bas. *Using High-Dimensional Image Models to Perform Highly Undetectable Steganography*. In: Information Hiding 2010. 2010.

[22] T. Pevny, P. Bas, and J. Fridrich. *Steganalysis by Subtractive Pixel Adjacency Matrix*. In: Information Forensics and Security, IEEE Transactions on (2010). ISSN: 1556-6013.

[23] Jessica J Fridrich and Jan Kodovsky. *Multivariate gaussian model for designing additive distortion for steganography*. In: ICASSP. 2013.

[24] Sarra Kouider, Marc Chaumont, and William Puech. *Adaptive steganography by oracle (ASO)*. In: Multimedia and Expo (ICME), 2013 IEEE International Conference on. IEEE. 2013.

[25] Vojtěch Holub, Jessica Fridrich, and Tomáš Denemark. *Universal distortion function for steganography in an arbitrary domain*. In: EURASIP Journal on Information Security (2014).

[26] Tomáš Denemark and Jessica Fridrich. *Side-informed steganography with additive distortion*. In: Information Forensics and Security (WIFS), 2015 IEEE International Workshop on. IEEE. 2015.

[27] Bin Li, Ming Wang, Jiwu Huang, et al. *A new cost function for spatial image steganography*. In: Image Processing (ICIP), 2014 IEEE International Conference on. IEEE. 2014.

[28] Bin Li, Ming Wang, Xiaolong Li, et al. *A strategy of clustering modification directions in spatial image steganography*. In: Information Forensics and Security, IEEE Transactions on (2015).

[29] Linjie Guo, Jiangqun Ni, Wenkang Su, et al. *Using Statistical Image Model for JPEG Steganography: Uniform Embedding Revisited*. In: IEEE Transactions on Information Forensics and Security (2015).

[30] Vahid Sedighi, Rémi Cogranne, and Jessica Fridrich. *Content-Adaptive Steganography by Minimizing Statistical Detectability*. In: Information Forensics and Security, IEEE Transactions on (2016).

[31] Weixuan Tang, Bin Li, Shunquan Tan, et al. *CNN-Based Adversarial Embedding for Image Steganography*. In: IEEE Transactions on Information Forensics and Security (2019).

[32] Jessica Fridrich and Jan Kodovsky. *Rich models for steganalysis of digital images*. In: Information Forensics and Security, IEEE Transactions on (2012).

[33] Vojtěch Holub and Jessica Fridrich. *Low-complexity features for JPEG steganalysis using undecimated DCT*. In: IEEE Transactions on Information Forensics and Security (2015).

[34] Xiaofeng Song, Fenlin Liu, Chunfang Yang, et al. *Steganalysis of adaptive JPEG steganography using 2D Gabor filters*. In: Proceedings of the 3rd ACM workshop on information hiding and multimedia security. ACM. 2015.

[35] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[36] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, et al. *What is the best multi-stage architecture for object recognition?* In: 2009 IEEE 12th International Conference on Computer Vision. 2009.

[37] Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright. *Optimization for Machine Learning*. Neural information processing series. MIT Press, 2012. ISBN: 9780262016469.

[38] Yassine Yousfi, Jan Butora, Eugene Khvedchenya, et al. *ImageNet pre-trained CNNs for JPEG steganalysis*. In: 2020 IEEE International Workshop on Information Forensics and Security (WIFS). IEEE. 2020.

[39] Yinlong Qian, Jing Dong, Wei Wang, et al. *Deep learning for steganalysis via convolutional neural networks*. In: IS&T/SPIE Electronic Imaging. International Society for Optics and Photonics. 2015.

[40] Patrcik Bas, Tomáš Pevny, and Tomáš Filler. *BossBase*. http://exile.felk.cvut.cz/boss. 2011.

[41] Guanshuo Xu. *Deep convolutional neural network to detect J-UNIWARD*. In: Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security. 2017.

[42] Sergey Ioffe and Christian Szegedy. *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. In: International conference on machine learning. PMLR. 2015.

[43] Mehdi Boroumand, Mo Chen, and Jessica Fridrich. *Deep residual network for steganalysis of digital images*. In: IEEE Transactions on Information Forensics and Security (2018).

[44] Rémi Cogranne, Quentin Giboulot, and Patrick Bas. *ALASKA-2 : Challenging Academic Research on Steganalysis with Realistic Images*. In: IEEE International Workshop on Information Forensics and Security. New York City (Virtual Conference), United States, 2020.

[45] Mingxing Tan and Quoc Le. *Efficientnet: Rethinking model scaling for convolutional neural networks*. In: International Conference on Machine Learning. PMLR. 2019.

[46] Jia Deng, Wei Dong, Richard Socher, et al. *Imagenet: A large-scale hierarchical image database*. In: 2009 IEEE conference on computer vision and pattern recognition. Ieee. 2009.

[47] Yassine Yousfi, Jan Butora, Jessica Fridrich, et al. *Improving EfficientNet for JPEG Steganalysis*. In: Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security. 2021.

[48] Patrick Bas and Teddy Furon. *BOWS-2*. http://bows2.ec-lille.fr. 2007.

[49] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, et al. *Intriguing properties of neural networks*. In: arXiv preprint arXiv:1312.6199 (2013).

[50] Weixuan Tang, Bin Li, Shunquan Tan, et al. *CNN-based Adversarial Embedding for Image Steganography*. In: IEEE Transactions on Information Forensics and Security (2019).

[51] Tomas Pevny and Andrew D Ker. *Exploring non-additive distortion in steganography*. In: Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security. 2018.

[52] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[53] Andrew D Ker and Tomas Pevny. *The steganographer is the outlier: realistic large-scale steganalysis*. In: (2014).

[54] Rémi Cogranne, Quentin Giboulot, and Patrick Bas. *The ALASKA Steganalysis Challenge: A First Step Towards Steganalysis "Into The Wild"*. In: ACM IH&MMSec (Information Hiding & Multimedia Security). ACM IH&MMSec (Information Hiding & Multimedia Security). Paris, France, 2019.

[55] Ulisse Dini. *Fondamenti per la teoria delle funzioni di variabli reali [Foundations of the theory of functions of real variable]*. Pisa, 1878.

[56] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. *Generative adversarial nets*. In: Advances in Neural Information Processing Systems. 2014.

[57] Jianhua Yang, Danyang Ruan, Jiwu Huang, et al. *An embedding cost learning framework using GAN*. In: IEEE Transactions on Information Forensics and Security (2019).

[58] Frans A Oliehoek, Rahul Savani, Jose Gallego-Posada, et al. *GANGs: Generative adversarial network games*. In: arXiv preprint arXiv:1712.00679 (2017).

[59] Lionel Pibre, Jérôme Pasquet, Dino Ienco, et al. *Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover sourcemismatch*. In: Electronic Imaging. Society for Imaging Science and Technology, 2016.

[60] J. Ye, J. Ni, and Y. Yi. *Deep Learning Hierarchical Representations for Image Steganalysis*. In: (2017).

[61] Guanshuo Xu, Han-Zhou Wu, and Yun-Qing Shi. *Structural design of convolutional neural networks for steganalysis*. In: IEEE Signal Processing Letters (2016).

[62] Patrick Bas, Tomáš Filler, and Tomáš Pevny. *"Break Our Steganographic System": The Ins and Outs of Organizing BOSS*. In: Information Hiding 2011. Lecture Notes in Computer Science. Czech Republic, 2011.

[63] Martin Abadi, Paul Barham, Jianmin Chen, et al. *Tensorflow: A system for large-scale machine learning*. In: 2016.

[64] Rémi Cogranne, Vahid Sedighi, Jessica Fridrich, et al. *Is ensemble classifier needed for steganalysis in high-dimensional feature spaces?* In: Information Forensics and Security (WIFS), 2015 IEEE International Workshop on. IEEE. 2015.

[65] Yassine Yousfi, Jan Butora, Jessica Fridrich, et al. *Breaking ALASKA: Color separation for steganalysis in JPEG domain*. In: Proceedings of the ACM Workshop on Information Hiding and Multimedia Security. 2019.

[66] Tomáš Filler, Jan Judas, and Jessica Fridrich. *Minimizing embedding impact in steganography using trellis-coded quantization*. In: Media forensics and security II. International Society for Optics and Photonics. 2010.

[67] Jessica Fridrich and Tomáš Filler. *Practical methods for minimizing embedding impact in steganography*. In: Security, Steganography, and Watermarking of Multimedia Contents IX. International Society for Optics and Photonics. 2007.

[68] Christy Kin-Cleaves and Andrew D Ker. *Simulating Suboptimal Steganographic Embedding*. In: Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security. 2020.

[69] Eric Jang, Shixiang Gu, and Ben Poole. *Categorical Reparameterization with Gumbel-Softmax*. In: 2017. URL: https://arxiv.org/abs/1611.01144.

[70] Eric Jang, Shixiang Gu, and Ben Poole. *Categorical reparameterization with gumbel-softmax*. In: arXiv preprint arXiv:1611.01144 (2016).

[71] Patrick Bas, Tomáš Filler, and Tomáš Pevny. *"Break Our Steganographic System": The Ins and Outs of Organizing BOSS*. In: International Workshop on Information Hiding. Springer Berlin Heidelberg, 2011.

[72] Diederik P Kingma and Jimmy Ba. *Adam: A method for stochastic optimization*. In: arXiv preprint arXiv:1412.6980 (2014).

[73] Vojtěch Holub, Jessica Fridrich, and Tomáš Denemark. *Universal distortion function for steganography in an arbitrary domain*. In: EURASIP Journal on Information Security (2014).

[74] Jan Butora and Jessica Fridrich. *Reverse JPEG compatibility attack*. In: IEEE Transactions on Information Forensics and Security (2019).

[75] Théo Taburet, Patrick Bas, Wadih Sawaya, et al. *Natural Steganography in JPEG Domain With a Linear Development Pipeline*. In: IEEE Transactions on Information Forensics and Security (2021).

[76] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, et al. *Towards deep learning models resistant to adversarial attacks*. In: arXiv preprint arXiv:1706.06083 (2017).

Computation of the gradient

A.1 The gradient of the modification w.r.t. probabilities

We want to compute $\frac{\partial \tilde{b}_k}{\partial \pi_i^j}$ with $\tilde{b}_k = \sum_{l \in \mathcal{B}} l z_k^l$, where $z_k^l = \frac{e^{\frac{g_k^l + \log \pi_k^l}{\tau}}}{\sum_{m \in \mathcal{B}} e^{\frac{g_k^m + \log \pi_k^m}{\tau}}}$.

Firstly,

$$\begin{aligned}\frac{\partial e^{\frac{g_k^l + \log \pi_k^l}{\tau}}}{\partial \pi_i^j} &= \frac{1}{\tau} \frac{\partial \log \pi_k^l}{\partial \pi_i^j} e^{\frac{g_k^l + \log \pi_k^l}{\tau}} \\ &= \frac{1}{\tau} [i = k][j = l] \frac{1}{\pi_i^j} e^{\frac{g_i^j + \log \pi_i^j}{\tau}}\end{aligned}$$

So

$$\begin{aligned}\frac{\partial z_k^l}{\partial \pi_i^j} &= \frac{\left(\sum_{m \in \mathcal{B}} e^{\frac{g_k^m + \log \pi_k^m}{\tau}}\right) \frac{\partial e^{\frac{g_k^l + \log \pi_k^l}{\tau}}}{\partial \pi_i^j} - \left(\sum_{m \in \mathcal{B}} \frac{\partial e^{\frac{g_k^m + \log \pi_k^m}{\tau}}}{\partial \pi_i^j}\right) e^{\frac{g_k^l + \log \pi_k^l}{\tau}}}{\left(\sum_{m \in \mathcal{B}} e^{\frac{g_k^m + \log \pi_k^m}{\tau}}\right)^2} \\ &= \frac{\left(\sum_{m \in \mathcal{B}} e^{\frac{g_k^m + \log \pi_k^m}{\tau}}\right) \frac{1}{\tau}[i = k][j = l] \frac{1}{\pi_i^j} e^{\frac{g_i^j + \log \pi_i^j}{\tau}} - \left(\sum_{m \in \mathcal{B}} \frac{1}{\tau}[i = k][j = m] \frac{1}{\pi_i^j} e^{\frac{g_i^j + \log \pi_i^j}{\tau}}\right) e^{\frac{g_k^l + \log \pi_k^l}{\tau}}}{\left(\sum_{m \in \mathcal{B}} e^{\frac{g_k^m + \log \pi_k^m}{\tau}}\right)^2} \\ &= \frac{1}{\tau}[i = k] \left([j = l] \frac{1}{\pi_i^j} z_i^j - \frac{1}{\pi_i^j} z_i^j z_k^l\right) \\ &= \frac{1}{\tau}[i = k] \frac{1}{\pi_i^j} z_i^j \left([j = l] - z_k^l\right)\end{aligned}$$

Finally:

$$\begin{aligned}\frac{\partial \tilde{b}_k}{\partial \pi_i^j} &= \sum_{l \in \mathcal{B}} l \frac{\partial z_k^l}{\partial \pi_i^j} \\ &= \sum_{l \in \mathcal{B}} l \frac{1}{\tau}[i = k] \frac{1}{\pi_i^j} z_i^j \left([j = l] - z_k^l\right) \\ &= \frac{1}{\tau}[i = k] \frac{z_i^j}{\pi_i^j} \sum_{l \in \mathcal{B}} l \left([j = l] - z_k^l\right) \\ &= \frac{1}{\tau}[i = k] \frac{z_i^j}{\pi_i^j} \left(j - \tilde{b}_i\right)\end{aligned}$$

A.2 The gradient fo the probabilities w.r.t. costs

We want to compute $\frac{\partial \pi_i^j}{\partial \rho_k^l}$, with $\pi_i^j = \frac{e^{-\lambda \rho_i^j}}{\sum_{m \in \mathcal{B}} e^{-\lambda \rho_i^m}}$.

Firstly, we have :

$$\frac{\partial e^{-\lambda \rho_i^j}}{\partial \rho_k^l} = -\lambda \frac{\rho_i^j}{\rho_k^l} e^{-\lambda \rho_i^j} = -\lambda [i=k][j=l] e^{-\lambda \rho_i^j}$$

So

$$\begin{aligned}
\frac{\partial \pi_i^j}{\partial \rho_k^l} &= \frac{\left(\sum_{m \in \mathcal{B}} e^{-\lambda \rho_i^m}\right)\left(\frac{\partial e^{-\lambda \rho_i^j}}{\partial \rho_k^l}\right) - \left(\sum_{m \in \mathcal{B}} \frac{\partial e^{-\lambda \rho_i^m}}{\partial \rho_k^l}\right)\left(e^{-\lambda \rho_i^j}\right)}{\left(\sum_{m \in \mathcal{B}} e^{-\lambda \rho_i^m}\right)^2} \\
&= \frac{\left(\sum_{m \in \mathcal{B}} e^{-\lambda \rho_i^m}\right)\left(-\lambda [i=k][j=l] e^{-\lambda \rho_i^j}\right) - \left(\sum_{m \in \mathcal{B}} -\lambda [i=k][m=l] e^{-\lambda \rho_i^m}\right)\left(e^{-\lambda \rho_i^j}\right)}{\left(\sum_{m \in \mathcal{B}} e^{-\lambda \rho_i^m}\right)^2} \\
&= -\lambda [i=k] \frac{e^{-\lambda \rho_i^j}}{\sum_{m \in \mathcal{B}} e^{-\lambda \rho_i^m}} \left([j=l] \frac{\sum_{m \in \mathcal{B}} e^{-\lambda \rho_i^m}}{\sum_{m \in \mathcal{B}} e^{-\lambda \rho_i^m}} - \frac{e^{-\lambda \rho_i^j}}{\sum_{m \in \mathcal{B}} e^{-\lambda \rho_i^m}} \right) \\
&= \lambda [i=k] \pi_i^j \left(\pi_i^j - [j=l] \right)
\end{aligned}$$

A.3 The gradient of the probabilities w.r.t. λ

Computing $\frac{\partial \pi_i^j}{\partial \lambda}$, with $\pi_i^j = \frac{e^{-\lambda \rho_i^j}}{\sum_{m \in \mathcal{B}} e^{-\lambda \rho_i^m}}$.

Firstly

$$\frac{\partial e^{-\lambda \rho_i^j}}{\partial \lambda} = -\rho_i^j e^{-\lambda \rho_i^j}$$

So

$$\begin{aligned}
\frac{\partial \pi_i^j}{\partial \lambda} &= \frac{\left(\sum_{m \in \mathcal{B}} e^{-\lambda \rho_i^m}\right)\left(-\rho_i^j e^{-\lambda \rho_i^j}\right) - \left(\sum_{m \in \mathcal{B}} -\rho_i^m e^{-\lambda \rho_i^m}\right) e^{-\lambda \rho_i^j}}{\left(\sum_{m \in \mathcal{B}} e^{-\lambda \rho_i^m}\right)^2} \\
&= -\rho_i^j \frac{e^{-\lambda \rho_i^j}}{\sum_{m \in \mathcal{B}} e^{-\lambda \rho_i^m}} + \frac{e^{-\lambda \rho_i^j}}{\sum_{m \in \mathcal{B}} e^{-\lambda \rho_i^m}} \frac{\sum_{m \in \mathcal{B}} \rho_i^m e^{-\lambda \rho_i^m}}{\sum_{m \in \mathcal{B}} e^{-\lambda \rho_i^m}} \\
&= -\rho_i^j \pi_i^j + \pi_i^j \sum_{m \in \mathcal{B}} \rho_i^m \pi_i^m \\
&= \pi_i^j \left(\sum_{m \in \mathcal{B}} \rho_i^m \pi_i^m - \rho_i^j \right)
\end{aligned}$$

A.4 The gradient of the entropy w.r.t. λ

We want to compute $\frac{\partial H(\boldsymbol{\pi})}{\partial \lambda}$, with $H(\boldsymbol{\pi}) = -\sum_{i=1}^{N} \sum_{j \in \mathcal{B}} \pi_i^j \log \pi_i^j$.

$$\begin{aligned}\frac{\partial H(\boldsymbol{\pi})}{\partial \lambda} &= -\sum_{i=1}^{N}\sum_{j\in\mathcal{B}}\left(\frac{\partial \pi_i^j}{\partial \lambda}\log \pi_i^j + \pi_i^j \frac{\partial \pi_i^j}{\partial \lambda}\frac{1}{\pi_i^j}\right)\\ &= -\sum_{i=1}^{N}\sum_{j\in\mathcal{B}}\frac{\partial \pi_i^j}{\partial \lambda}\left(\log \pi_i^j + 1\right)\end{aligned}$$

A.5 The gradient of the entropy w.r.t. costs

We want to compute $\dfrac{\partial H(\boldsymbol{\pi})}{\partial \rho_k^l}$, with $H(\boldsymbol{\pi}) = -\displaystyle\sum_{i=1}^{N}\sum_{j\in\mathcal{B}}\pi_i^j \log \pi_i^j$.

$$\begin{aligned}\frac{\partial H(\boldsymbol{\pi})}{\partial \rho_k^l} &= -\sum_{i=1}^{N}\sum_{j\in\mathcal{B}}\left(\frac{\partial \pi_i^j}{\partial \rho_k^l}\log \pi_i^j + \pi_i^j \frac{\partial \pi_i^j}{\partial \rho_k^l}\frac{1}{\pi_i^j}\right)\\ &= -\sum_{i=1}^{N}\sum_{j\in\mathcal{B}}\frac{\partial \pi_i^j}{\partial \rho_k^l}\left(\log \pi_i^j + 1\right)\\ &= -\sum_{i=1}^{N}\sum_{j\in\mathcal{B}}\lambda[i=k]\pi_i^j\left(\pi_i^j - [j=l]\right)\left(\log \pi_i^j + 1\right)\\ &= \lambda\sum_{j\in\mathcal{B}}\pi_k^j\left([j=l] - \pi_k^j\right)\left(\log \pi_k^j + 1\right)\end{aligned}$$

Vizualisation of various costs maps

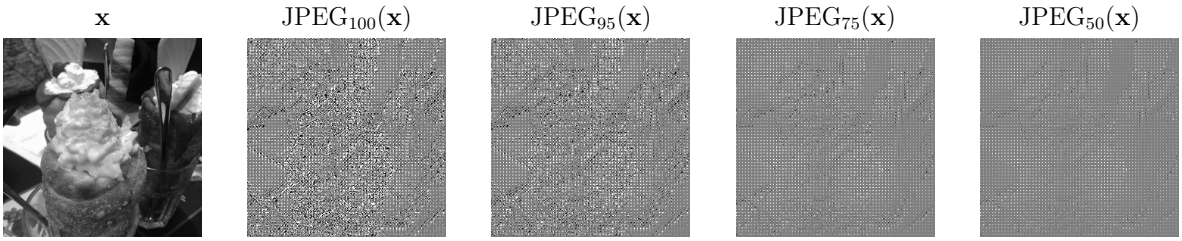


Figure 1: Original image

B.6 In spatial domain

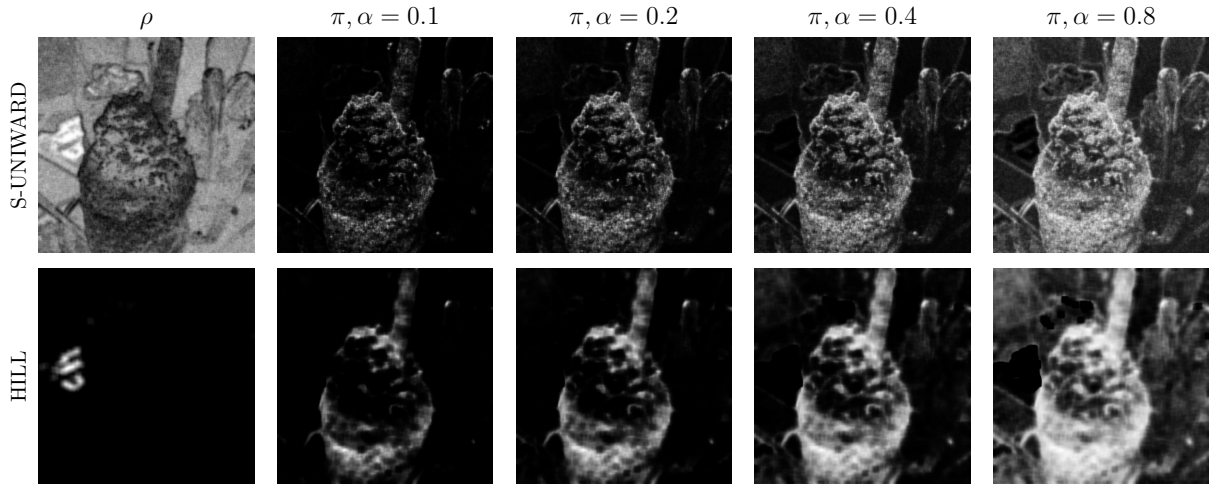


Figure 2: Spatial embedding: S-UNIWARD and HILL cost maps

B.7 In JPEG domain

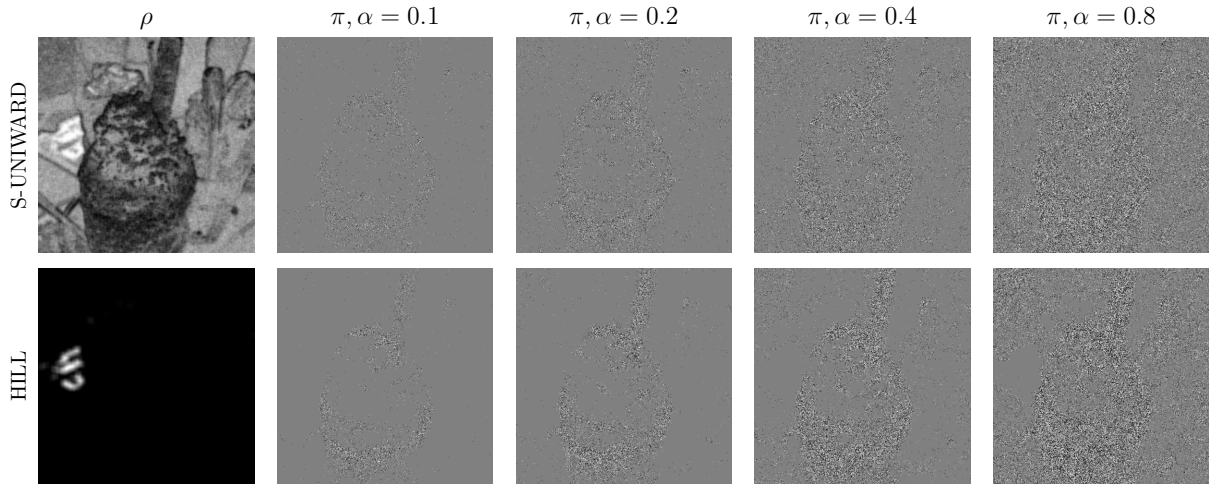


Figure 3: Spatial embedding: S-UNIWARD and HILL embedding changes

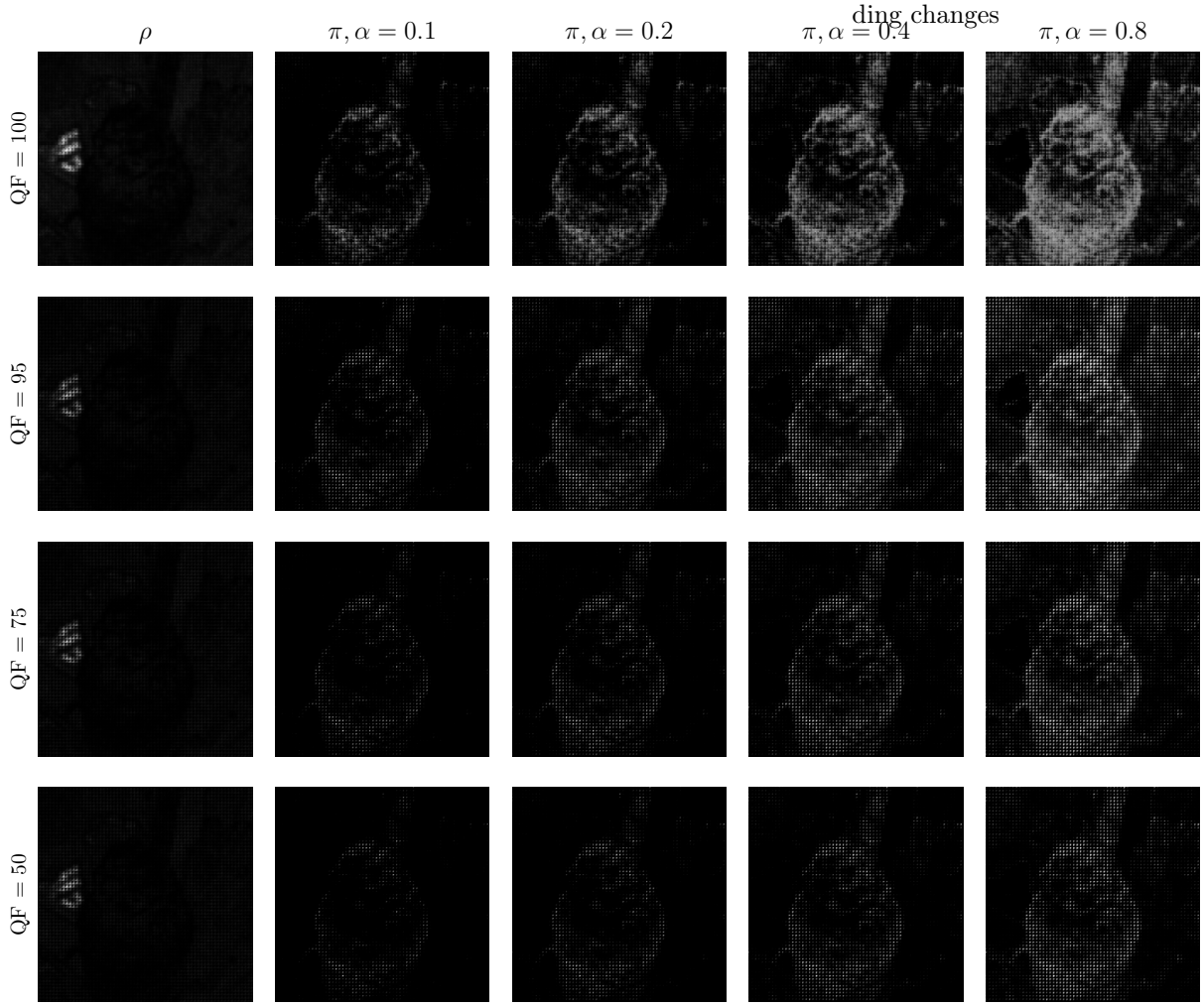


Figure 4: J-UNIWARD cost maps at different QF, and probability maps at different relative payload per non-zero AC coefficient

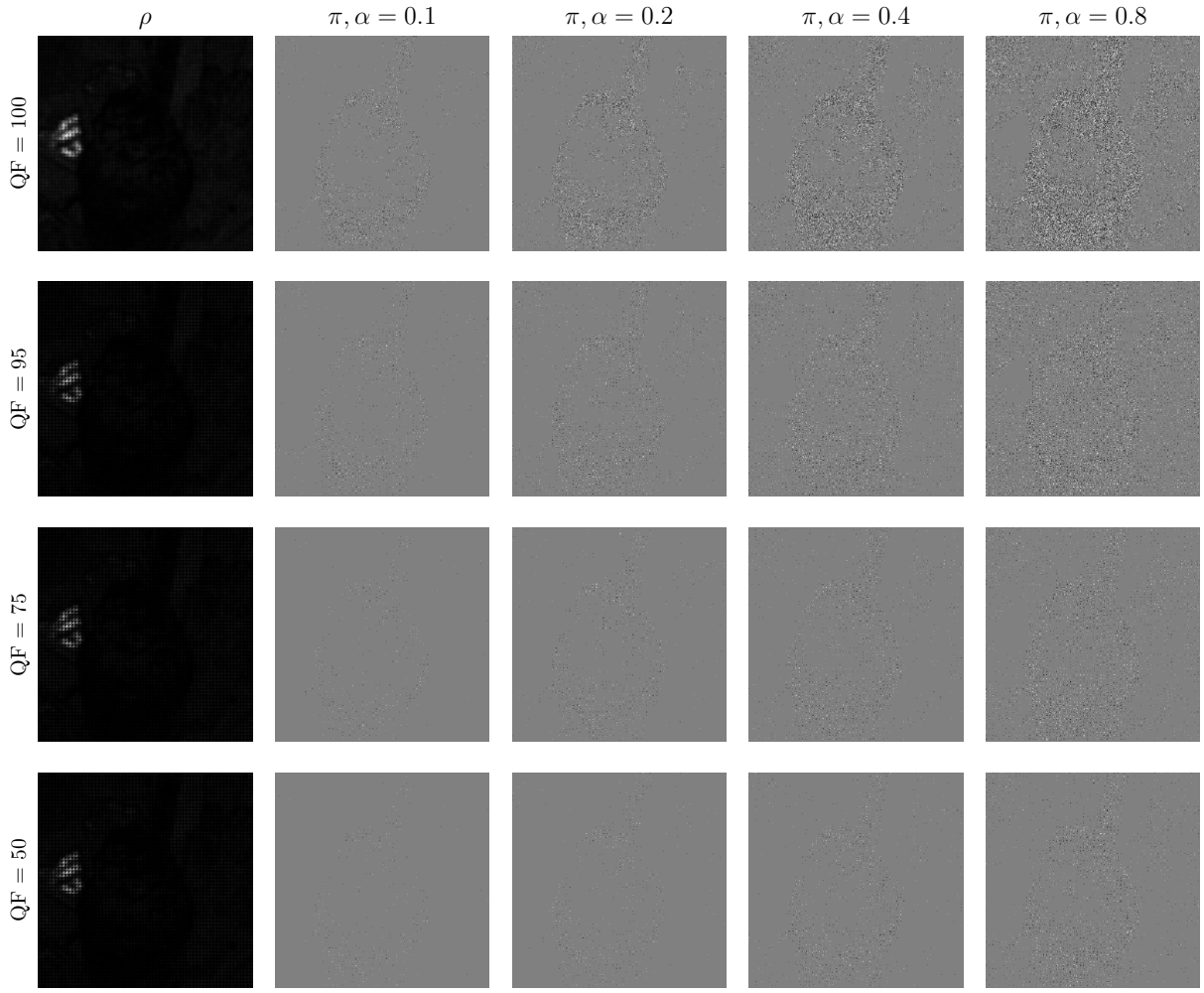


Figure 5: J-UNIWARD cost maps at different QF, and probability maps at different relative payload per non-zero AC coefficient

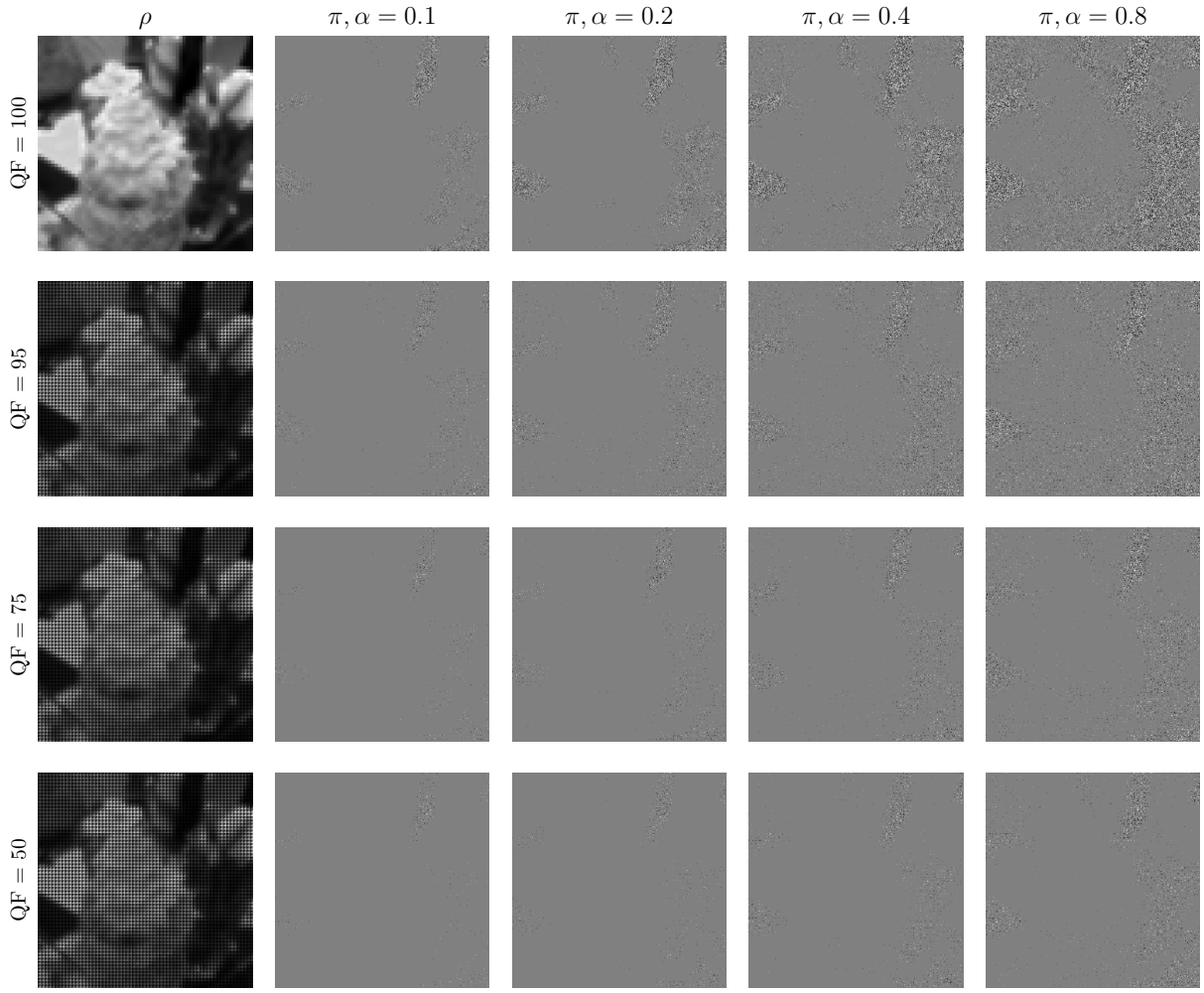


Figure 6: UERD cost maps at different QF, and probability maps at different relative payload per non-zero AC coefficient

Abstract

Steganography is a technique that Alice can use to send a message to Bob secretly. It consists of concealing a message discreetly within a public content without arousing the steganalyst Eve's suspicion, who observes the communication channel and can cut it. Digital images offer an enormous potential to hide a message by modifying its coefficient slightly. In this thesis, we present an automatic procedure to improve steganography by cover modification.

To increase security, Alice should adapt the embedding to the content to avoid risky areas, such as the sky or walls. Her mission consists, therefore, of designing a distortion function defining which coefficients should not be changed. On the other hand, Eve can use state-of-the-art detectors by training a deep learning model to detect which image was modified. They are very performant to detect invisible-to-the-eye modifications made to a cover.

In this manuscript, we propose using game theory notions to tackle the problem, as we can see Alice and Eve playing a competitive game with antagonistic ambitions. It helps us define the problem correctly and highlights the min max optimization problem Alice wants to solve.

Our first contribution proposes a practical solution to solve the optimization problem by constructing a fictional iterative game where the two players can play optimally. First, (i) Alice can create new stegos by adapting her distortion function to defeat Eve's optimal differentiable detector. Then (ii) Eve can create a new classifier optimized to detect Alice's best stegos. In that way, we propose an automatic procedure that improves Alice's scheme w.r.t a differentiable detector. We used for our experiments the algorithm ADV-EMB [31] for step (i). The results show that the automatic protocol gives stegos undetectable by a specific detector and even unexpected classifiers.

Our second contribution proposes an improvement of ADV-EMB. When its solutions rely on a heuristic, we propose to adopt a classical optimization technique, gradient descent, to approach the resolution of the optimization problem. The key idea was to apply a re-parametrization trick that allows differentiating the expectation of the detectability given by a detector with respect to the distortion function. This new method provides a more powerful attack, which again improves Alice's discretion.

Résumé en français

La stéganographie est une technique qu'Alice peut utiliser pour envoyer secrètement un message à Bob. Cela consiste à dissimuler discrètement un message au sein d'un contenu public sans éveiller les soupçons du stéganalyste Eve, qui observe le canal de communication et peut le couper. Par exemple, écrire à l'encre invisible est une technique de stégaganographie, et chauffer une feuille pour vérifier la présence d'un message est une technique de stéganalyse. Les images numériques, d'une part de leur la banalité de leur présence sur internet et d'autre part pour leur structure toujours compliquée à modéliser, offrent un potentiel énorme pour cacher un message en modifiant légèrement ses coefficients. Dans cette thèse, nous présentons une procédure automatique pour améliorer la stéganographie en modifiant des images naturelles.

D'une manière générale, l'état-de-l'art conseille, pour augmenter la sécurité, qu'Alice doit adapter l'incorportation du message au contenu afin d'éviter les zones à risque, comme le ciel ou les murs. Sa mission consiste donc à concevoir une fonction de distorsion définissant quels coefficients de l'image ne doivent pas être modifiés. Pour simplifier les calculs, la fonction de distortion est souvent définie comme une somme de coûts de modifications indépendants. C'est-à-dire que à chaque coefficient i de l'image est associé un coût ρ^b correspondant à l'impact de la modification du coefficient par l'ajout de la valeur b.

L'additivité des coûts dans la définition de la distortion permet d'utiliser l'algorithme efficace Syndrome Treillis Code (STC) pour cacher n'importe quel message dans une image. En théorie, cette méthode basée sur de la programmation linéaire, permet d'atteindre la borne minimale de distortion tout en cachant le message, cela produit ainsi une image stégo optimale. Seulement cette méthode, bien que produisant une méthode efficace par rapport à la complexité du problème, reste coûteuse à exécuter. C'est pourquoi en pratique l'état-de-l'art suggère de simuler l'insertion du message. Avec une longueur d'un message fictif donné, nous pouvons calculer la distribution de probabilité des modifications à appliquer à l'image pour cacher ce message tout en minimisant parfaitement la distortion. Ainsi à chaque pixel est associé une distribution de probabilité discrète de modification ; simuler l'insertion revient à tirer au hasard une modification selon cette distribution. Les paramètres de la distribution s'obtiennent par une formule directe des coûts de modification ρ, et d'un scalaire λ defini implicitement pour satisfaire une condition sur l'entropie, devant être égale à la longueur du message à cacher.

D'autre part, Eve peut utiliser des détecteurs de pointe en entraînant un modèle d'apprentissage profond pour détecter quelles images

ont été modifiées. Ils sont très performants pour détecter des modifications invisibles à l'œil nu effectuées à une image. L'état-de-l'art a vu se développer plusieurs architectures adaptées à telles tâches, pour analyser les signaux faibles dans les images. Les architectures utilisées dans cette thèse seront XU-Net, SRNet et Efficient-Net.

Ainsi, Alice et Eve ont deux rôles opposés : Alice veut transmettre un message sans être détectée par Eve, alors que cette dernière veut détecter la présence d'un message dans les images transmises publiquement par Alice. Dans ce manuscrit, nous proposons d'utiliser des notions de théorie des jeux pour aborder le problème, car nous pouvons voir Alice et Eve comme jouant à un jeu compétitif avec des ambitions antagonistes. Cela nous aide à définir correctement le problème et met en évidence le problème d'optimisation min max qu'Alice veut résoudre. La première étape de maximisation se fait sur l'ensemble des actions d'Eve, supposant qu'elle choisit son meilleur classifieur, et la deuxième étape est une étape de minimisation sur l'ensemble des actions d'Alice, choisissant la meilleure réponse face à l'action d'Eve pour minimiser son taux de bonne classification.

Seulement ce problème d'optimisation s'avère être très compliqué à résoudre, à cause du nombre d'actions disponibles à chaque joueur. En effet, il existe une inifinité de possibilité pour les deux, ce qui rend la résolution du problème min max complexe.

Notre première contribution propose une solution pratique pour résoudre le problème d'optimisation en construisant un jeu itératif fictif où les deux joueurs peuvent jouer de manière optimale. A la place de résoudre ce jeu d'un coup, nous proposons de résoudre successivement plusieurs jeux de plus en plus difficiles, mais à chaque fois résolubles. A chaque iteration, chaque joueur propose une meilleure action, visant à la fin à améliorer la stéganographie vis-à-vis d'une architecture de stéganalyse.

Plus précisémement, tout d'abord, le jeu commence par la création de la part d'Alice d'un ensemble d'images stégos, et de la part d'Eve la création d'un steganalyste discriminant bien ces stégos des images naturelles covers. Puis chaque itération, chaque joueur va proposer une nouvelle action. D'abord (i) Alice peut créer de nouvelles stégos en adaptant sa fonction de distorsion pour vaincre le détecteur dérivable optimal d'Eve. Puis (ii) Eve peut créer un nouveau classificateur optimisé pour détecter les meilleurs stégos d'Alice. De cette façon, nous proposons une procédure automatique qui améliore le schéma d'Alice par rapport à un détecteur dérivable. Nous avons utilisé pour nos expériences l'algorithme ADV-EMB [31] pour l'étape (i). Les résultats montrent que le protocole automatique donne des stégos indétectables par un détecteur spécifique mais aussi par des classifieurs non considérés jusqu'alors.

Notre deuxième contribution propose une amélioration de ADV-EMB. Le but d'une attaque de la part d'Alice consiste à minimiser l'espérance de la détéctabilité du meilleur classifieur d'Eve par rapport à la fonction de distortion (ie les coûts de modification additifs). Nous observons que c'est ce qu'effectue ADV-EMB, mais est limité

au bout de quelques itérations du protocole de notre première contribution, ne fournissant plus une attaque assez performante. De plus, cet algorithme est basé sur une heuristique. Nous proposons, dans notre deuxième contribution, d'adopter pour aborder la résolution du problème d'optimisation une technique d'optimisation classique, la descente de gradient, afin de minimiser l'espérance de la détectabilité, tout en enlevant le paramètre heuristique de ADV-EMB. Nous suspectons que cette méthode n'était pas appliquée jusqu'alors à cause d'une difficulté encontrée lors du calcul du gradient. En effet, il nécessite de calculer en outre le gradient d'un tirage aléatoire discret par rapport aux paramètres de la distribution. La reparamétrisation du problème permet d'avoir une expression formelle du gradient, mais à cause de la discretisation du tirage, le gradient a une valeur soit nulle soit non définie. Ainsi, l'idée principale est d'approximer la fonction de discretisation par une fonction dérivable, contrôlée par une valeur λ. Ainsi, nous pouvons finalement différencier l'espérance de la détectabilité donnée par un détecteur par rapport à la fonction de distorsion. Cette nouvelle méthode fournit une attaque plus puissante, qui améliore encore plus la discrétion d'Alice.

Digital images steganography using adversarial embedding

Steganography is a technique Alice can use to secretly send a message to Bob. This consists of discreetly concealing a message within public content without arousing the suspicion of the steganalyst Eve, who observes the communication channel. In this thesis, we present an automatic procedure to improve steganography via natural image modification. To increase security, Alice must adapt the embedding of the message to the content by designing a distortion function defining which coefficients of the image must not be modified. On the other hand, Eve can use advanced detectors by training a deep learning model to detect which images have been altered. In this manuscript, we propose to use notions of game theory to approach the problem, because we can see Alice and Eve as playing a competitive game with antagonistic ambitions. This helps us to define the problem correctly and highlights the min max optimization problem that Alice wants to solve. Our first contribution proposes a practical solution to solve the optimization problem by building a fictitious iterative game where both players can play optimally. First, (*i*) Alice can create new stegos by adapting her distortion function to defeat Eve's optimal derivable detector. Then (*ii*) Eve can create a new classifier optimized to detect Alice's best stegos. In this way, we propose an automatic procedure which improves Alice's scheme compared to a derivable detector. The results show that the automatic protocol gives stegos undetectable by a specific detector but also by classifiers not considered until now. Our second contribution proposes an improvement of ADV-EMB previously used for the (*i*) step. While this algorithm is based on a heuristic, we propose to adopt a classical optimization technique, gradient descent, to approach the resolution of the optimization problem. The main idea is to apply reparametrize the sampling such that it is possible to differentiate the expectation of the detectability given by a detector with respect to the distortion function. This new method provides a more powerful attack, which further improves Alice's discretion.
Keywords: Security, Steganography, Steganalysis

Stéganographie d'images numériques via l'utilisation de réseaux de neurones sous présence d'un adversaire

La stéganographie est une technique qu'Alice peut utiliser pour envoyer secrètement un message à Bob. Cela consiste à dissimuler discrètement un message au sein d'un contenu public sans éveiller les soupçons du stéganalyste Eve, qui observe le canal de communication. Dans cette thèse, nous présentons une procédure automatique pour améliorer la stéganographie en modifiant des images naturelles. Pour augmenter la sécurité, Alice doit adapter l'incorportation du message au contenu en concevant une fonction de distorsion définissant quels coefficients de l'image ne doivent pas être modifiés. D'autre part, Eve peut utiliser des détecteurs de pointe en entraînant un modèle d'apprentissage profond pour détecter quelles images ont été modifiées. Dans ce manuscrit, nous proposons d'utiliser des notions de théorie des jeux pour aborder le problème, car nous pouvons voir Alice et Eve comme jouant à un jeu compétitif avec des ambitions antagonistes. Cela nous aide à définir correctement le problème et met en évidence le problème d'optimisation min max qu'Alice veut résoudre. Notre première contribution propose une solution pratique pour résoudre le problème d'optimisation en construisant un jeu itératif fictif où les deux joueurs peuvent jouer de manière optimale. Tout d'abord, (*i*) Alice peut créer de nouvelles stégos en adaptant sa fonction de distorsion pour vaincre le détecteur dérivable optimal d'Eve. Puis (*ii*) Eve peut créer un nouveau classificateur optimisé pour détecter les meilleurs stégos d'Alice. De cette façon, nous proposons une procédure automatique qui améliore le schéma d'Alice par rapport à un détecteur dérivable. Les résultats montrent que le protocole automatique donne des stégos indétectables par un détecteur spécifique mais aussi par des classifieurs non considérés jusqu'alors. Notre deuxième contribution propose une amélioration de ADV-EMB jusqu'alors utilisé pour l'étape (*i*). Alors que cet algorithme repose sur une heuristique, nous proposons d'adopter une technique d'optimisation classique, la descente de gradient, pour aborder la résolution du problème d'optimisation. L'idée principale est de reparamétriser l'étape d'échantillonage de sorte que l'on puisse différencier l'espérance de la détectabilité donnée par un détecteur par rapport à la fonction de distorsion. Cette nouvelle méthode fournit une attaque plus puissante, qui améliore encore plus la discrétion d'Alice.
Mots clés : Sécurité, Stéganographie, Stéganalyse