HAL
open science

Deep Learning in steganography and steganalysis

Marc Chaumont

To cite this version:

Marc Chaumont. Deep Learning in steganography and steganalysis. M. Hassaballah. Digital Media Steganography: Principles, Algorithms, Advances, Elsevier, pp.321-349, 2020, Chapter 14 Deep Learning in steganography and steganalysis, 978-0-12-819439-3. 10.1016/B978-0-12-819438-6.00022-0 . lirmm-02087729v2

HAL Id: lirmm-02087729

https://hal-lirmm.ccsd.cnrs.fr/lirmm-02087729v2

Submitted on 16 Oct 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

(final version - October 2019). Elsevier Book chapter, Elsevier Inc, Book titled: "*Digital Media Steganography: Principles, Algorithms, Advances*", Book Editor: M. Hassaballah, 2020.

CHAPTER

Deep Learning in steganography and steganalysis from 2015 to 2018

1

Marc CHAUMONT[a,*]

[*] *Montpellier University, LIRMM (UMR5506) / CNRS, Nîmes University, France, LIRMM/ICAR, 161, rue Ada, 34392 Montpellier, France*

[a] *Corresponding:* `marc.chaumont@lirmm.fr`

ABSTRACT

For almost 10 years, the detection of a hidden message in an image has been mainly carried out by the computation of Rich Models (RM), followed by classification using an Ensemble Classifier (EC). In 2015, the first study using a convolutional neural network (CNN) obtained the first results of steganalysis by Deep Learning approaching the performances of the two-step approach (EC + RM). Between 2015-2018, numerous publications have shown that it is possible to obtain improved performances, notably in spatial steganalysis, JPEG steganalysis, Selection-Channel-Aware steganalysis, and in quantitative steganalysis. This chapter deals with deep learning in steganalysis from the point of view of current methods, by presenting different neural networks from the period 2015-2018, that have been evaluated with a methodology specific to the discipline of steganalysis. The chapter is not intended to repeat the basic concepts of machine learning or deep learning. So, we will present the structure of a deep neural network, in a generic way and present the networks proposed in existing literature for the different scenarios of steganalysis, and finally, we will discuss steganography by deep learning.

Keywords: Steganography, steganalysis, Deep Learning, GAN

Neural networks have been studied since the 1950s. Initially, they were pro-

posed to model the behavior of the brain. In computer science, especially in artificial intelligence, they have been used for around 30 years for learning purposes. Ten or so years ago [37], neural networks were considered to have a lengthy learning time and to be less effective than classifiers such as SVMs or random forests.

With recent advances in the field of neuron networks [6], thanks to the computing power provided by graphics cards (GPUs), and because of the profusion of available data, deep learning approaches have been proposed as a natural extension of neural networks. Since 2012, these deep networks have profoundly marked the fields of signal processing and artificial intelligence, because their performances make it possible to surpass current methods, but also to solve problems that scientists had not managed to solve until now[60].

In steganalysis, for the last 10 years, the detection of a hidden message in an image was mainly carried out by calculating Rich Models (RM) [28] followed by a classification using a classifier (EC) [51]. In 2015, the first study using a convolutional neural network (CNN) obtained the first results of deep-learning steganalysis approaching the performances of the two-step approach (EC + RM [1]) [80]. During the period 2015 - 2018, many publications have shown that it is possible to an obtain improved performance in spatial steganalysis, JPEG steganalysis, side-informed steganalysis, quantitative steganalysis, etc.

In Section 1.1 we present the structure of deep neural network generically. This Section is centered on existing publications in steganalysis and should be supplemented by reading about artificial learning and in particular gradient descent, and stochastic gradient descent. In Section 1.2 we explain the different steps of the convolution module. In Section 1.3 we will tackle the complexity and learning times. In Section 1.4 we will present the links between Deep Learning and previous approaches. In Section 1.5 we will revisit the different networks that were proposed during the period 2015-2018 for different scenarios of steganalysis. Finally, in Section 1.6 we will discuss steganography by deep learning which sets up a game between two networks in the manner of the precursor algorithm ASO [57].

1.1 THE BUILDING BLOCKS OF A DEEP NEURONAL NETWORK

In the following sub-sections, we look back at the major concepts of a Convolutional Neural Network (CNN). But specifically, we will recall the basic

[1] We will note EC + RM in order to indicate the two-step approach based on the calculation of Rich Models (RM) then the use of an ensemble classifier (EC).

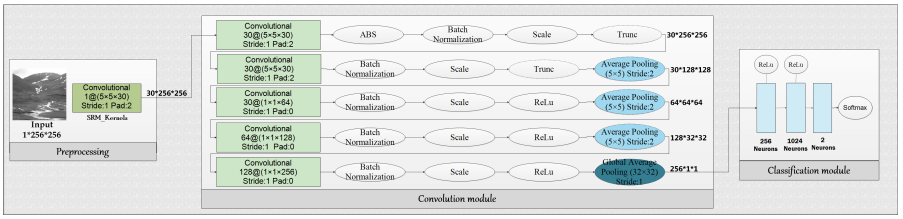

Figure 1.1 Yedroudj-Net network [108].

building blocks of a network based on the Yedroudj-Net[2] network that was published in 2018 [108] (See Figure 1.1), and which takes up the ideas present in Alex-Net [58], as well as the concepts present in networks developed for steganalysis including the very first network of Qian *et al.* [80], and networks of Xu-Net [101], and Ye-Net [106].

1.1.1 GLOBAL VIEW OF A CONVOLUTIONAL NEURAL NETWORK

Before describing the structure of a neural network as well as its elementary components, it is useful to remember that a neural network belongs to the machine-learning family. In the case of supervised learning, which is the case that most concerns us, it is necessary to have a database of images, with, for each image, its label, that is to say, its class.

Deep Learning networks are large neural networks that can directly take raw input data. In image processing, the network is directly powered by the pixels forming the image. Therefore, a deep learning network learns in a joint way, both the compact intrinsic characteristics of an image (we speak of *feature map* or of *latent space*) and at the same time, the separation boundary allowing the classification (we also talk of *separator plans*).

The learning protocol is similar to classical machine learning methods. Each image is given as input to the network. Each pixel value is transmitted to one or more neurons. The network consists of a given number of *blocks*. A block consists of neurons that take real input values, perform calculations, and then transmit the actual calculated values to the next block. A neural network can, therefore, be represented by an oriented graph where each node represents a computing unit. The learning is then completed by supplying the network with examples composed of an image and its label, and the network modifies the parameters of these calculation units (it learns) thanks to the mechanism of back-propagation.

[2] GitHub link on Yedroudj-Net: `https://github.com/yedmed/steganalysis_with_CNN_Yedroudj-Net`.

The Convolutional Neuronal Networks used for steganalysis are mainly built in three parts, which we will call *modules*: the pre-processing module, the convolution module, and the classification module. As an illustration, figure 1.1 schematizes the network proposed by Yedroudj *et al.* in 2018 [108]. The network processes grayscale images of 256×256 pixels.

1.1.2 THE PRE-PROCESSING MODULE

$$F^{(0)} = \frac{1}{12}\begin{pmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{pmatrix} \tag{1.1}$$

We can observe in Figure 1.1 that in the *pre-processing module*, the image is filtered by 30 high-pass filters. The use of one or more high-pass filters as pre-processing is present in the majority of networks used for steganalysis during the period 2015-2018.

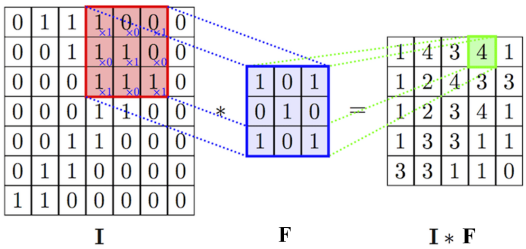

Figure 1.2 Principle of a convolution.

An example of a kernel of a high-pass filter – the square S5a filter [28] – is given in Equation 1.1. An illustration of the filtering (convolution) principle is given in Figure 1.2. This preliminary filtering step allows the network to converge faster and is probably needed to obtain good performance when the learning database is too small [107] (only 4 000 pairs cover/stego images of size 256×256 pixels). The filtered images are then transmitted to the first convolution block of the network. Note that the recent SRNet [10] network does not use any fixed pre-filters, but learns the filters. It therefore requires a much larger database (more than 15 000 pairs cover/stego images of size 256×256 pixels), and strong know-how for its initialization. Note that there is a debate in the community if one should use fixed filters, or initialize the filters with pre-chosen values and then continue the learning, or learn filters with random initialization. At the beginning of 2019, in practice (real-world situation [48]), the best choice is probably in relation to the size of the learning

database (which is not necessary BOSS [4] or BOWS2 [3]), and the possibility to use transfer learning.

1.1.3 THE CONVOLUTION MODULE

Within the *convolution module*, we find several macroscopic computation units that we will call *blocks*. A *block* is composed of calculation units that take real input values, perform calculations, and return real values, which are supplied to the next block. Specifically, a *block* takes a set of *feature maps* (= a set of images) as input and returns a set of *feature maps* as output (= a set of images). Inside a block, there are a number of operations including the following four: the *convolution* (see Section 1.2.1), the *activation* (see Section 1.2.2), the *pooling* (see Section 1.2.3), and finally the *normalization* (see Section 1.2.4).

Note that the concept of neuron, as defined in existing literature, before the emergence of convolutional networks, is still present, but it no longer exists as a data structure in neural network libraries. In convolution modules, we must imagine a neuron as a computing unit which, for a position in the *feature map* taken by the convolution kernel during the convolution operation, performs the weighted sum between the kernel and the group of considered pixels. The concept of neuron corresponds to the scalar product between the input data (the pixels) and data specific to the neuron (the weight of the convolution kernel), followed by the application of a function of $\mathbb{R}$ in $\mathbb{R}$ called the activation function. Then, by extension, we can consider that pooling and normalization are operations specific to neurons.

Thus, the notion of *block* corresponds conceptually to a " layer " of neurons. Note that in deep learning libraries, we call a *layer* any elementary operation such as convolution, activation, pooling, normalization, etc. To remove any ambiguity, for the convolution module we will talk about *block*, and *operations*, and we will avoid using the term *layer*.

Without counting the pre-processing block, the *Yedroudj-Net* network [108] has a convolution module made of 5 convolution blocks, like the networks of Qian *et al.* [80] and Xu *et al.* [101]. The *Ye-Net* network [106] has a convolution module composed of 8 convolution blocks, and SRNet network [10] has a convolution module built with 11 convolution blocks.

1.1.4 THE CLASSIFICATION MODULE

The last block of the convolution module (see the previous Section) is connected to the *classification module* which is usually a *fully connected* neural network composed of one to three blocks. This *classification module* is often a traditional neural network where each neuron is fully connected to the previous *block* of neurons and to the next *block* of neurons.

The fully connected blocks often end with a softmax function which normalize the outputs delivered by the network between $[0, 1]$, such that the sum

of the outputs equal one. The outputs are named imprecisely "probabilities". We will keep this denomination. So, in the usual binary steganalysis scenario, the network delivers two values as output: one giving the probability of classifying into the first class (e.g. the cover class), and the other giving the probability of classifying into the second class (e.g. the stego class). The classification decision is then obtained by returning the class with the highest probability.

Note that in front of this *classification module*, we can find a *particular pooling* operation such as a *global average pooling*, a *Spatial Pyramid Pooling (SPP)* [34], a *statistical moments extractor* [97], etc. Such pooling operations return a fixed-size vector of values, that is to say, a feature map of fixed dimensions. The next block to this *pooling* operation is thus always connected to a vector of fixed size. So, this block has a fixed input number of parameters. It is thus possible to present to the network images of any size without having to modify the topology of the network. For example, this property is available in the Yedroudj-Net [108] network, the Zhu-Net [115] network, or the Tsang et al. network [97].

Also note that [97] is the only paper, at the time of writing this chapter, which has seriously considered the viability of an invariant network to the dimension of the input images. The problem remains open. The solution proposed in [97] is a variant of the concept of average pooling. For the moment, there has not been enough studies on the subject to determine what is the correct topology of the network, how to build the learning data-base, how much the number of embedded bits influences the learning, or if we should take into account the *square root law* for learning at a fixed security-level or any payload size, etc.

1.2 THE DIFFERENT STEPS OF THE CONVOLUTION MODULE

In Section 1.1.3, we indicated that a block within the convolution module contained a variable number among the following four operations: the *convolution* (see Section 1.2.1), the *activation* (see Section 1.2.2), the *pooling* (see Section1.2.3), and finally the *normalization* (see Section 1.2.4). Let's now explain in more detail each step (convolution, activation, pooling, and normalization) within a *block*.

1.2.1 CONVOLUTION

The first treatment within a *block* is often to apply the convolutions on the input *feature maps*.

Note that for the pre-processing *block*, see Figure 1.1, there is only one input

image. A convolution is therefore carried out between the input image and a filter. In the Yedroudj-Net network, there are 30 high-pass filters extracted from SRM filters [28]. In older networks, there is only one pre-processing filter [80, 78, 101].

Except for the pre-processing *block*, in the other *blocks*, once the convolution has been applied, we apply activation steps (see Section 1.2.2), pooling (see Section 1.2.3), and normalization (see Section 1.2.4). Then we obtain a new image named *feature map*.

Formally, let $I^{(0)}$ be the input image of the pre-processing *block*. Let $F_k^{(l)}$ be the kth ($k \in \{1, ..., K^{(l)}\}$) filter of the *block* of number $l = \{1, ..., L\}$, with L the number of *blocks*, and with $K^{(l)}$ the number of filters of the lth *block*. The convolution within the pre-processing *block* with the kth filter results in a filtered image, denoted $\tilde{I}_k^{(1)}$, such that:

$$\tilde{I}_k^{(1)} = I^{(0)} \star F_k^{(1)}. \tag{1.2}$$

From the first *block of the convolution module* to the last *block* of convolution (see Figure 1.1), the convolution is less conventional because there is $K^{(l-1)}$ *feature maps* ($K^{(l-1)}$ images) as input, denoted $I_k^{(l-1)}$ with $k = \{1, ..., K^{(l-1)}\}$.

The "convolution" that will lead to the kth filtered image, $\tilde{I}_k^{(l)}$, resulting from the convolution *block* numbered l, is actually the sum of $K^{(l-1)}$ convolutions, such as:

$$\tilde{I}_k^{(l)} = \sum_{i=1}^{i=K^{(l-1)}} I_i^{(l-1)} \star F_{k,i}^{(l)}, \tag{1.3}$$

with $\{F_{k,i}^{(l)}\}_{i=1}^{i=K^{(l-1)}}$ a set of $K^{(l-1)}$ filters for a given k value.

This operation is quite unusual since each *feature map* is obtained by a *sum* of $K^{(l-1)}$ convolutions with a different filter kernel for each convolution. This operation can be seen as a spatial convolution, plus a sum on the channels-axis [3].

This combined operation can be replaced by a separate operation called *SeparableConv* or *Depthwise Separable Convolutions* [16], which allows us to integrate a non-linear operation (an activation function) such as a ReLU, between the spatial convolution and the convolution on the "depth" axis (for the "depth" axis we use a 1×1 filter). Thus, the *Depthwise Separable Convolution* can roughly be resumed as a weighted sum of convolution which is a more descriptive operation than just a sum of convolution (see Equation 1.3).

If we replace the operation previously described in equation 1.3, by a *Depth-*

[3] The channels axis is also referred by "feature maps"-axis, or "depth"-axis.

wise Separable Convolutions operation integrated within an *Inception* module (the Inception allows us to mainly use filters of variable sizes), we obtain a performance improvement [16]. In steganalysis, this has been observed in the article [115], when modifying the first two blocks of the convolution module of Figure 1.1.

As a reminder, in this document, we name a *convolution block* the set of operations made by one convolution (or many convolutions performed in parallel in the case of an Inception, and/or two convolutions in the case of a Depthwise Separable convolution), a few activation functions, a pooling, and a normalization. These steps can be formally expressed in a simplified way (except in cases with Inception or Depthwise Separable Convolution) in recursive form by linking a *feature map* at the input of a block and the *feature map* at the output of this block:

$$I_k^{(l)} = norm\left(pool\left(f\left(b_k^{(l)} + \sum_{i=1}^{i=K^{(l-1)}} I_i^{(l-1)} \star F_{k,i}^{(l)}\right)\right)\right), \tag{1.4}$$

with $b_k^{(l)} \in \mathbb{R}$ the scalar standing for the convolution bias, $f()$ the activation function applied pixel by pixel on the filtered image, $pool()$, the *pooling* function that is applied to a local neighborhood, and finally a normalization function.

Note that the kernels of the filters (also called weights) and the bias must be learned and are therefore modified during the back-propagation phase.

1.2.2 ACTIVATION

Once each convolution of a *convolution block* has been applied, an *activation* function, $f()$ (see Eq. 1.4), is applied on each value of the filtered image, $\tilde{I}_k^{(l)}$ (Eq. 1.2 and Eq. 1.3). This function is called the activation function with reference to the notion of binary activation found in the very first work on neuron networks. The activation function can be one of several, for example be an absolute value function $f(x) = |x|$, a sinusoidal function $f(x) = sinus(x)$, a Gaussian function as in [80] $f(x) = \frac{e^{-x^2}}{\sigma^2}$, a ReLU (for *Rectified Linear Unit*): $f(x) = max(0, x)$, etc. Figure 1.3 illustrates some activation functions.

These functions break the linearity resulting from linear filtering performed during convolutions. Non-linearity is a mandatory property that is also exploited in *two-step machine-learning approaches*, such as in the ensemble classifier [51] during the weak-classifiers thresholding, or through the final majority vote, or in Rich Models with Min-Max features [28]. The chosen activation function must be differentiable to perform back-propagation.

The most often retained solution for the selection of an activation function is one whose derivative requires little calculation to be evaluated. Besides, functions that have low slope regions, such as the hyperbolic tangent, are

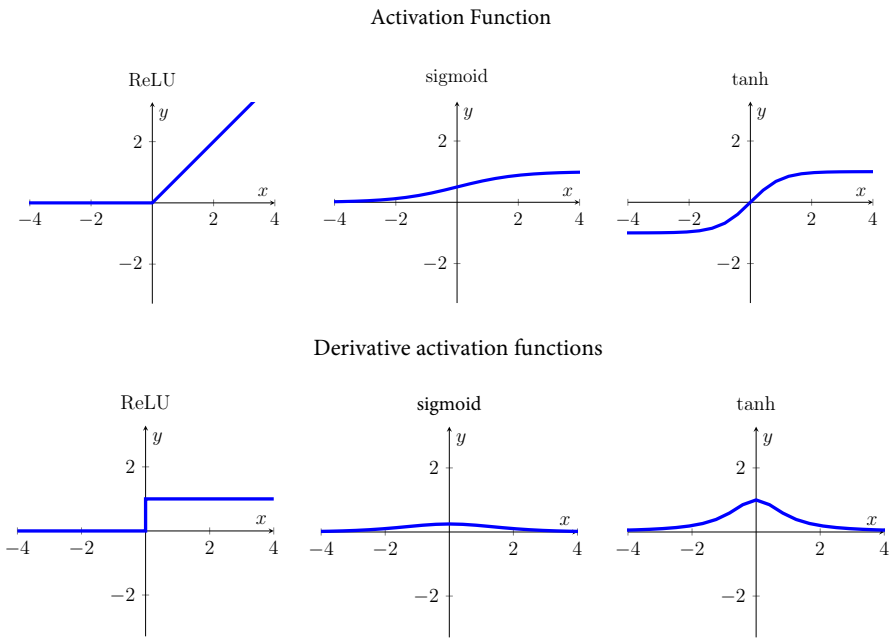

Figure 1.3 Three main activation functions and their derivatives.

also avoided, since this type of function can cause the value of the back-propagated gradient to be canceled during back-propagation (the phenomenon of the *vanishing gradient*), and thus will make learning impossible. Therefore, in many networks, we very often find the ReLU activation function, or one of its variants. For example, in the Yedroudj-Net network (see figure 1.1) we have the absolute value function, the parameterized Hard Tanh function (Trunc function), and the ReLU function. In the SRNet network [10] we only find the ReLU function.

1.2.3 POOLING

The pooling operation is used to calculate the *average* or the *maximum* in a local neighborhood. In the field of classification of objects in images, the maximum pooling guarantees a local invariance in translation when recomputing the features. That said, in most steganalysis networks, it is preferred to use average pooling to preserve stego noise which is of very low power. Figure 1.4 illustrates the two pooling operations.

Moreover, pooling is often coupled to a down-sampling operation (when the *stride* is greater than 1) to reduce the size (i.e., the height and width) of the resulting *feature map* compared to feature maps from the previous block.

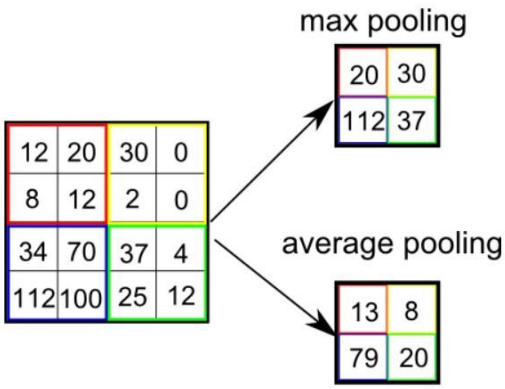

Figure 1.4 Illustration of a maximum pooling and an average pooling.

For example, in Yedroudj-Net (see figure 1.1), blocks 2, 3, and 4, reduce by a four-factor the size of the input feature maps. We can consider the pooling operation, accompanied by a stride greater than 1, as conventional sub-sampling with preliminary low-pass filtering. This is useful for reducing the amount of used-memory in the GPU. This step can also be perceived as denoising, and from the point of view of the signal processing, it induces a loss of information. It is probably better not to sub-sample in the first blocks as it was initially highlighted in [78], set up in Xu-Net [101], Ye-Net [106], Yedroudj-Net [108], and evaluated again in SRNet [10].

1.2.4 NORMALIZATION

In the first proposed networks in steganalysis, during the period 2014 − *beginning of* 2016 (Tan and Li [94], Qian *et al.* [80], Pibre *and al.* [78]), if there was a normalization, it remained local to the spatial neighborhood, with *Local Constrast Normalization*, or inter-feature, with the *Local Response Normalization*.

A big improvement occurred with the arrival of *batch normalisation*. *Batch normalization* (BN) was proposed in 2015 [47], and was widely adopted. This normalization is present in most of the new networks for steganalysis. BN [47] (see Eq. 1.5) consists of normalizing the distribution of each feature of a feature map, so that the average is zero and the variance is unitary, and, if necessary allows re-scaling and re-translating the distribution.

Given a random variable X whose realization is a value $x \in \mathbb{R}$ of the feature map, the BN of this value x is:

$$BN(x, \gamma, \beta) = \beta + \gamma \frac{x - E[X]}{\sqrt{Var[X] + \epsilon}}, \tag{1.5}$$

with $E[X]$ the expectation, $Var[X]$ the variance, and γ and β two scalars representing a re-scaling and a re-translation. The expectation $E[X]$ and the variance $Var[X]$ are updated at each batch, while γ and β are learned by back-propagation. In practice, the BN makes the learning less sensitive to the initialization of parameters [47], allows us to use a higher learning rate which speeds up the learning process, and improves the accuracy of classification [14].

In Yedroudj-Net, the terms γ and β are treated by an independent layer called *Scale Layer* (See Figure 1.1), in the same way as in ResNet [35]. The increment in performance is very minor.

1.3 MEMORY / TIME COMPLEXITY AND EFFICIENCY

Learning a network can be considered as the optimization of a function with many unknown parameters, thanks to the use of a well-thought out stochastic gradient descent. In the same way as traditional neural networks, the CNNs used for steganalysis have a large number of parameters to learn. As an example, without taking into account Batch Normalization and Scale parameters, the Xu-Net [101] network described in the paper [108] has a number of parameters in the order of 50,000. In comparison, the network Yedroudj-Net [108], has a number of unknown parameters in the order of 500,000.

In practice, using a previous-generation GPU (Nvidia TitanX) on an Intel Core i7-5930K at 3.50 GHz $\times$ 12 with 32 GB of RAM, it takes less than a day to learn the Yedroudj-Net network using 4,000 pairs of 256×256 cover/stego images of the " BOSS" [4], three days on 14,000 pairs of 256×256 cover/stego images of "BOSS + BOWS2" [3], and more than seven days on the 112,000 pairs of 256×256 cover/stego images of "BOSS + BOWS2 + a virtual database augmentation" [107]. These long learning times are because the databases are large and have to be browsed repeatedly, so that the back-propagation process makes converge the network.

Due to the large number of parameters to be learned, neural networks need a database containing a large number of examples to be in *the power-law region* [36] allowing comparisons between different networks. In addition, the examples within the learning database must be sufficiently diversified to obtain a good generalization of the network. For CNN steganalysis, with current networks (in 2018), the number of examples needed to reach a region of *good performance* (that is, as good as using a Rich Model[28] with an Ensemble Classifier [51]), in the case where there is no cover-source mismatch, is most likely in the order of 10,000 images (5,000 covers and 5,000 stegos) when the size is 256×256 pixels [107]. However, the number of examples is still insufficient [107] in the sense that performance can be increased simply by

increasing the number of examples. The so-called *irreducible error region* [36] probably requires more than a million images [113]; Therefore, there should be at least 100 times more images for the learning phase. In addition to this, it is necessary to be able to work with larger images. It is therefore evident that in the future it will be essential to find one or more solutions to reach the region of *irreducible error*. This can be done with huge databases, and several weeks or months of apprenticeships, or by transfer learning, by using better networks, or with solutions yet to be conceived.

Note that of course, there are recommendations to increase performance and it may be possible to achieve the *irreducible error* region faster. We can use transfer learning [79] and/or curriculum learning [106] to start learning from a network that has already learned. We can use a set of CNNs [103], or a network made of sub-networks [62], which can save a few percentage points on accuracy. We can virtually increase the database [58], but this does not solve the problem of increasing the learning time. We can add images of a database that is similar to the test database, for example when BOSS and BOWS2 are used for learning, in the case where the test is realized on BOSS [106], [107]. It is nevertheless not obvious that in practice we can have access to a database similar to the database to be tested. We can (i) predict the acquisition devices that produced the images of the test database, then (ii) make new acquisitions with these devices (to be purchased), and (iii) finally perform images development similar to the one used to generate the test database, and all this in order to increase the learning database [107]. Again, this approach is difficult to implement and time-consuming.

Note that a general rule shared by people playing with Kaggle competitions is that the main practical rules to win are [59] [4]: (i) to use an ensemble of modern networks (ResNet, DenseNet, etc.) that have learned for example on ImageNet, and then use transfer learning, and (ii) to do data-augmentation, (iii) to eventually collect data to increase the learning database size.

1.4 LINK BETWEEN DEEP-LEARNING AND PAST APPROACHES

In previous Sections, we explained that deep-learning consisted of minimizing a function with many unknown parameters with a technique similar to gradient descent. In this subsection, we establish links with previous research on the

[4] The authors of [59] finished second at the Kaggle competition for *IEEE's Signal Processing Society - Camera Model Identification - Identify from which camera an image was taken.*
`https://www.kaggle.com/c/sp-society-camera-model-identification`.
https://towardsdatascience.com/forensic-deep-learning-kaggle-camera-model-identification-challenge-f6a3892561bd.

subject in the steganography/steganalysis community. This sub-section tries to make links with past research in this domain and is an attempt to demystify deep learning.

Convolution is an essential part of CNN networks. Learning filter kernels (weights) are carried out by minimizing the classification error using the back-propagation procedure. It is, therefore, a simple optimization of filter kernels. Such a strategy can be found as early as 2012 in a two-step approach using Rich Models and an Ensemble Classifier in the article [38]. The kernel values used to calculate the feature vector are obtained by optimization via the simplex algorithm. In this article, the goal is to minimize the probability of classification error given by an Ensemble Classifier in the same way as with a CNN. CNNs share the same goal of building custom kernels that are well suited to steganalysis.

Looking at the first *block* of convolution just after the pre-processing *block* (Ye-Net [106], Yedroudj-Net [108], ReST-Net [62], etc.), the convolutions act as a multi-band filtering performed on the residuals obtained from the pre-processing block (see Figure 1.1). For this first block, the network analyzes the signal residue in different frequency bands. In the past, when computing Rich Models [28], some approaches have applied a similar idea thanks to the use of a filter bank. Some approaches make a spatio-frequency decomposition via the use of Gabor filters (GFR Rich Models) [92], [100], some use Discrete Cosinus filters (DCTR Rich Models) [41], some use Steerable Gaussian filters [2], and some make a projection on random carriers (PSRM Rich Models) [40], etc. For all these Rich Models, the results of the filtering process is then used to calculate a histogram (co-occurrence matrix) which is in turn used as a vector of features. The first convolution block of CNNs for steganalysis thus share similarities with the spatio-frequency decomposition of some Rich Models.

From the convolution blocks that start to down-sample the feature maps, there is a summation of the results of several different convolutions. This amounts to accumulating signs of the presence of a signal (the stego noise) by observing clues in several bands. We do not find such a principle in previous research. The only way to accumulate evidence was based on the computation of a histogram [28, 40], but this approach is different from what is done in CNNs. Note that in the article [86], the authors explore how to incorporate the histogram computation mechanism into a CNN network, but the results are not encouraging. Thus, starting from the second block, the mechanism involved to create a latent space separating the two classes, i.e. to obtain a feature vector per image, which makes it possible to distinguish the covers from the stegos, is different from that used in Rich Models. Similarly, some past techniques such as non-uniform quantization [74], features selection [13], dimension reduction [75], are not directly visible within a CNN.

A brick present in most convolution blocks is the normalization of feature

maps. Normalization has often been used in steganalysis, for example in [57], [17], [9], etc. Within a CNN, normalization is performed among other things to obtain comparable output values in each feature map.

The activation function introduces a non-linearity in the signal and thus makes it possible to have many convolution blocks. This non-linearity is found in the past, for exemple in the Ensemble Classifier through the majority vote [51], or in Rich Models with the Min or Max operations [28].

The structure of a CNN network and the components that improve the performance of a network are now better understood in practice. As we saw previously, there is in a CNN, some parts that are similar to propositions made in steganalysis in the past. Some elements of a CNN are also explained by the fact that they are guided by computational constraints (uses of simple differentiable activation function like ReLU), or that they facilitates the convergence (non-linearity allows convergence, activation function should not be too flat or steep, in order to avoid vanishing gradient or rapid variation, the shortcut allows us to avoid vanishing gradient during back-propagation, and thus allows us to create deeper networks, the batch normalization, the initialization such as Xavier, the optimization such as Adam, etc). Note that some of the ideas present in CNNs also come from the theory of optimization of differentiable functions.

Although it is easy to use a network in practice, and to have some intuition about its behavior, it still lacks theoretical justification. For example, what is the right number of parameters according to the problem? In the coming years, there is no doubt that the building of a CNN network adapted for steganalysis could go through an automatic adjustment of its topology, in this spirit, the work on AutoML and Progressive Neural Architecture Search (PNAS) [67], [77] are of interest. That said the theory must also try to explain what is happening inside the network. One can notably look at the work of Stéphane Mallat [70] for an attempt to explain a CNN from a signal processing point of view. Machine learning theorists can also better explain what happens in a network and why this mathematical construction works so well.

To conclude this discussion on the links between two-step learning approaches and deep learning approaches, CNN as well as two-step (Rich Models + Ensemble Classifier) approaches are not able to cope with cover-source mismatch [12, 29]. This is a defect used by detractors [5] of neural network approaches in domains such as object recognition [71]. CNNs learn a distribution, but if it differs in test phase, then the network cannot detect it. Maybe the ultimate goal is for the network to "understand" that the test database is not distributed as the learning database?

---

[5] See Gary Marcus' web-press article `https://medium.com/@GaryMarcus/the-deepest-problem-with-deep-learning-91c5991f5695`.

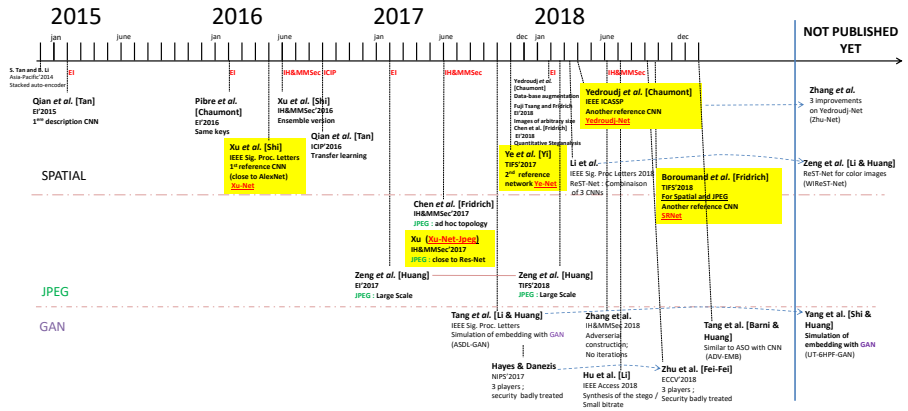

Figure 1.5 Chronology of the main CNNs for steganography and steganalysis from 2015 to 2018.

1.5 THE DIFFERENT NETWORKS USED OVER THE PERIOD 2015-2018

A chronology of the main CNNs proposed for steganography and steganalysis from 2015 to 2018 are given in Figure 1.5. The first attempt to use Deep Learning methods for steganalysis date back to the end of 2014 [94] with auto-encoders. At the beginning of 2015, Qian *et al.* [80] proposed to use Convolutional Neural Networks. One year later Pibre *et al.* [78] proposed to pursue the study.

In 2016, the first results, close to those of current state-of-the-art methods (Ensemble Classifier + Rich Models), were obtained with an ensemble of CNNs [103]; See Figure 1.6. The Xu-Net [6] [101] CNN is used as a *base learner* of an ensemble of CNNs.

Other networks were proposed in 2017, this time for JPEG steganalysis. In [114] [113] (See Figures 1.7 and 1.8), authors proposed a pre-processing inspired by Rich Models, and the use of a large learning database. The results were close to those of existing state-of-the-art methods (Ensemble Classifier + Rich Models). In [14], the network is built with a *phase-split* inspired by the JPEG compression process. An ensemble of CNNs was required to obtain results that were slightly better than those obtained by current best approach. In Xu-Net-Jpeg [102], a CNN inspired by ResNet [35] with the *shortcut connection* trick, and 20 blocks also improved the results in terms of

[6] In this chapter, we reference *Xu-Net* a CNN similar to the one given in [101], and not to the ensemble version [103].

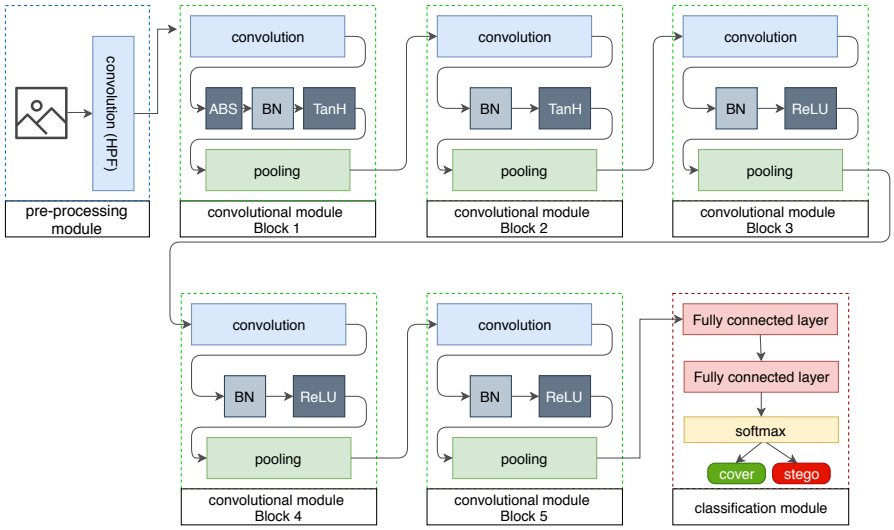

Figure 1.6 Xu-Net overall architecture.

accuracy. Note that in 2018 the ResDet [46] proposed a variant of **Xu-Net-Jpeg** [102] with similar results.

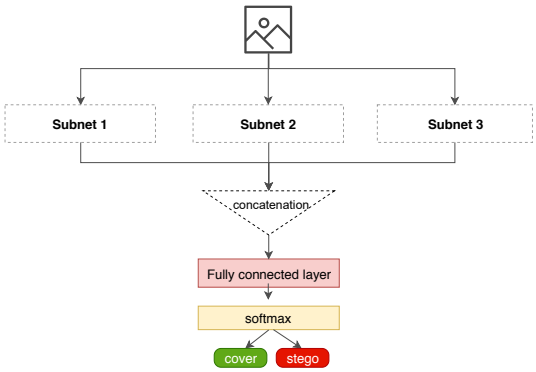

Figure 1.7 ReST-Net overall architecture.

These results were highly encouraging, but regarding the gain obtained in other image processing tasks using Deep Learning methods [60], the steganalysis results represented less than a 10% improvement compared to the classical approaches that use an Ensemble Classifier [51] with Rich Models [28], [99] or Rich Models with a Selection-Channel Awareness [20], [22], [21]. The revolutionary significant gain in the use of deep learning, observed in other areas

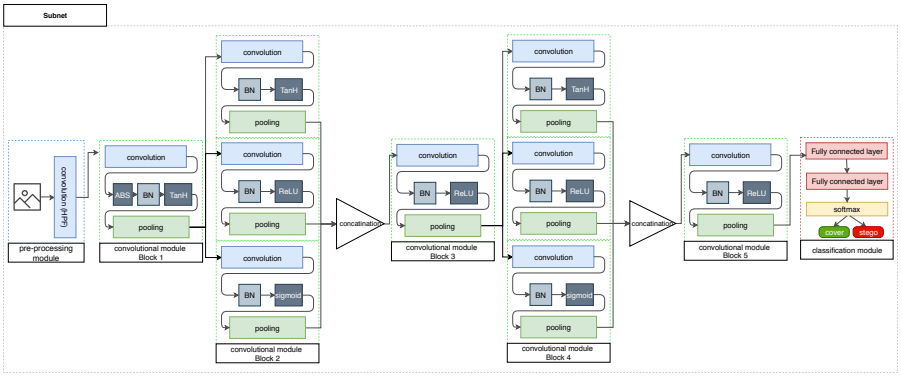

Figure 1.8 ReST-Net sub-network.

of signal processing, was not yet present for steganalysis. In 2017, the main trends to improve CNN results were using an ensemble of CNNs, modifying the topology by mimicking Rich Models extraction process or using ResNet. In most of the cases, the design or the experimental effort was very high for a very limited improvement of performance in comparison to networks such as AlexNet [58], VGG16 [91], GoogleNet [93], ResNet [35], etc, that inspired this research.

By the end of 2017 and early 2018, the studies had strongly concentrated on spatial steganalysis. Ye-Net [106] (See Figure 1.9), Yedroudj-Net[7] [107, 108] (See Figure 1.10), ReST-Net [62] (See Figures 1.7 and 1.8), SRNet[8] [10] (See Figures 1.11) have been published respectively in November 2017, January 2018, May 2018, and May 2019 (with an online version in September 2018). All these networks clearly surpass the "old" two-step machine learning paradigm that was using an Ensemble Classifier [51] and Rich Models [28]. Most of these networks can learn with a modest database size (i.e. around 15,000 pairs cover/stego of 8-bits-coded images of 256×256 pixels size from BOSS+BOWS2).

In 2018, the best networks were Yedroudj-Net [108], ReST-Net [62], and SRNet [10]. Yedroudj-Net is a small network that can learn on a very small database and can be effective even without using the tricks known to improve performance such as transfer learning [79] or virtual augmentation of the database [106], etc. This network is a good candidate when working on GANs. It is better than Ye-Net [106], and can be improved to face other more recent networks [115]. ReST-Net [62] is a huge network made of three sub-networks which uses various pre-processing filter banks. SRNet [10] is a

[7] Yedroudj-Net source code: `https://github.com/yedmed/steganalysis_with_CNN_Yedroudj-Net`.

[8] SRNet source code: `https://github.com/Steganalysis-CNN/residual-steganalysis`.

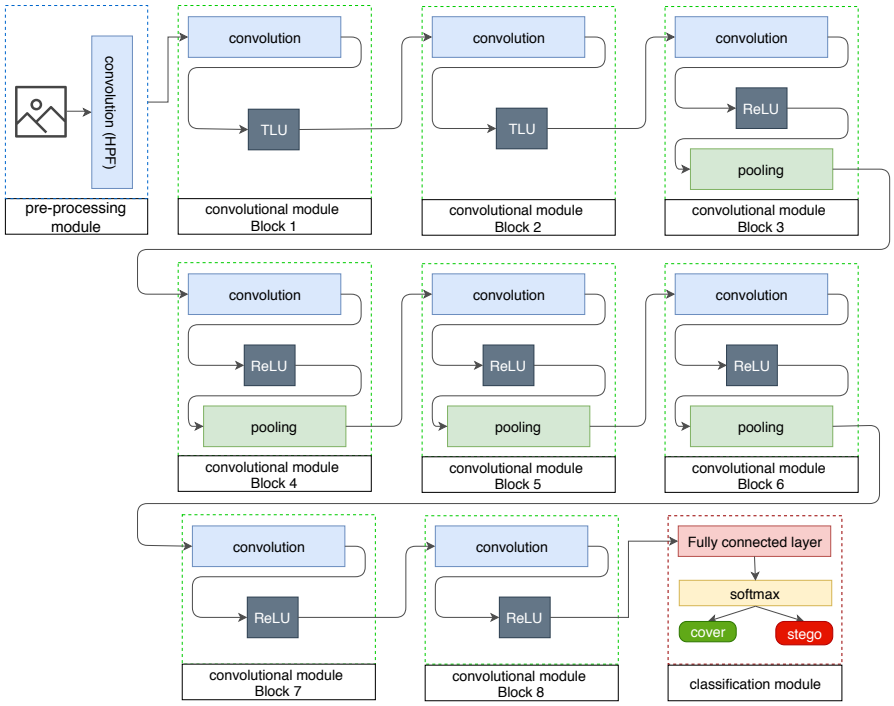

Figure 1.9 Ye-Net overall architecture.

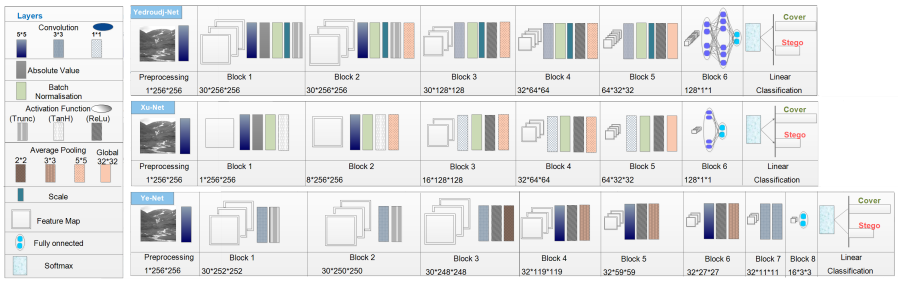

Figure 1.10 Comparison of Yedroudj-Net, Xu-Net, and Ye-Net architectures.

network that can be adapted to spatial or Jpeg steganalysis. It requires various tricks such as virtual augmentation and transfer learning, and therefore requires a bigger database compared to Yedroudj-Net. These three networks are described in Section 1.5.1.

To resume, from 2015 to 2016, publications were in spatial steganalysis, in 2017, the publications were mainly on JPEG steganalysis. In 2018, publications were again mainly concentrated on spatial steganalysis. Finally, at the

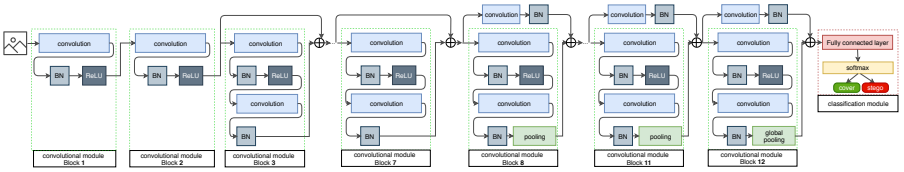

Figure 1.11 SRNet network.

end of 2017, the first publications using GANs appeared. In Section 1.6 we present the new propositions using steganography by deep-learning, and give classification per family.

In the next subsection, we report on the most successful networks until the end of 2018, for various scenarios. In Section 1.5.1, we describe the *Not-Side-Channel-Aware* (Not-SCA) scenario, in Section 1.5.2 we discuss the scenario known as *Side-Channel-Aware* (SCA), in Sections 1.5.3 we deal with JPEG steganalysis *Not-SCA* and *SCA* scenarios. In Section 1.5.4 we very briefly discuss cover-source mismatch, although for the moment the proposals using a CNN do not exist.

We will not tackle the scenario of CNN invariant to the size of the images because it is not yet mature enough. This scenario is briefly discussed in Section 1.1.4, and the papers of Yedroudj-Net [108], Zhu-Net [115], or Tsang *et al.* [97], give firsts solutions.

We will not approach the scenario of quantitative steganalysis per CNN, which consists in estimating the embedded payload size. This scenario is very well examined in the paper [15] and serves as a new state-of-the-art method. The approach surpasses the previous state-of-the-art approaches [53] [111] that rely on Rich Models, an Ensemble of trees, and an efficient normalization of features.

Nor will we discuss batch steganography and pooled steganalysis with CNNs which has not yet been addressed, although the work presented in [112] using two-stage machine learning can be extended to deep learning.

1.5.1 THE SPATIAL STEGANALYSIS NOT-SIDE-CHANNEL-AWARE (NOT-SCA)

In early 2018 the most successful spatial steganalysis approach is the Yedroudj-Net [108] method (See Figure 1.9). The experiments comparisons were carried out on the BOSS database which contains 10,000 images sub-sampled to 256×256 pixels. For a fair comparison, the experiments were performed by comparing the approach to Xu-Net without Ensemble [101], to the Ye-Net network in its Not-SCA version [106], and also to Ensemble Classifier [51] fed by Spatial-Rich-Models [28]. Note that Zhu-Net [115] (not yet published when writing this chapter) offers three improvements to Yedroudj-Net that allows

it to be even more efficient. The improvements reported by Zhu-Net [115] are the update to the kernel filters of the pre-processing module (in the same vein as what has been proposed by Matthew Stamm's team in Forensics [5]), replacing the first two convolution blocks with two modules of *Depthwise Separable Convolutions* as proposed in [16], and finally replacing global average pooling with a *Spatial Pyramid Pooling (SPP)* module as in [34].

In May 2018 the ReST-Net [62] approach was proposed (See Figures 1.7 and 1.8). It consists of agglomerating three networks to form a *super-network*. Each sub-net is a modified Xu-Net like network [101] resembling the Yedroudj-Net [108] network, with an Inception module on block 2 and block 4. This Inception module contains filters of the same size, with a different activation function for each "path" (TanH, ReLU, Sigmoid). The first subnet performs pre-processing with 16 Gabor filters, the second sub-network pre-processing with 16 SRM linear filters, and the third network pre-processing with 14 non-linear residuals (min and max calculated on SRM). The learning process requires four steps (one step per subnet and then one step for the *super-network*). The results are 2-5% better than Xu-Net for S-UNIWARD [43], HILL [63], CMD-HILL [64] on the BOSSBase v1.01 [4] 512 $\times$ 512. Looking at the results, it is the concept of Ensemble that improves the performances. Taken separately, each sub-net has a lower performance. At the moment, no comparison in a fair framework was made between an Ensemble of Yedroudj-Net and ReST-Net.

In September 2018 the SRNet [10] approach became available online (See Figures 1.11). It proposes a deeper network than previous networks, which is composed of 12 convolution blocks. The network does not perform pre-processing (the filters are learned) and sub-samples the signal only from the 8th convolution block. To avoid the problem of vanishing gradient, blocks 2 to 11 use the shortcut mechanism. The Inception mechanism is also implemented from block 8 during the pooling (sub-sampling) phase. The learning database is augmented with the BOWS2 database as in [106] or [107], and a curriculum training mechanism [106] is used to change from a standard payload size of 0.4 bpp to other payload sizes. Finally, gradient descent is performed by Adamax [49]. The network can be used for spatial steganalysis (Not-SCA), for informed (SCA) spatial steganalysis (see Section 1.5.2) and for JPEG steganalysis (see Section 1.5.3 Not-SCA or SCA). Overall the philosophy remains similar to previous networks, with three parts: pre-processing (with learned filters), convolution blocks, and classification blocks. With a simplified vision, the network corresponds to the addition of 5 blocks of convolution without pooling, just after the first convolution block of Yedroudj-Net network. To be able to use this large number of blocks on a modern GPU, authors must reduce the number of feature maps to 16, and in order to avoid the problem of vanishing gradients, they must use the trick of residual shortcut within the blocks as proposed in [35]. Note that preserving the size of the signal in the

first seven blocks is a radical approach. This idea has been put forward in [78] where the suppression of pooling had clearly improved the results. The use of modern brick like shortcuts or Inception modules also enhances performance.

It should also be noted that the training is completed end-to-end without particular initialization (except when there is a curriculum training mechanism). In the initial publication [10], SRNet network was not compared to Yedroudj-Net [108], or to Zhu-Net [115], but later, in 2019, in [115] all these networks have been compared and the update of Yedroudj-Net i.e. Zhu-Net gives performances of 1% to 4% improvement over SRNet, and 4% to 9% improvement over Yedroudj-Net, when using the usual comparison protocol. Note that Zhu-Net is also better than the network *Cov-Pool* published at IH&MMSec'2019 [23], and whose performances are similar to SRNet.

1.5.2 THE SPATIAL STEGANALYSIS SIDE-CHANNEL-INFORMED (SCA)

At the end of 2018, two approaches combined the knowledge of the selection channel, the SCA-Ye-Net (which is the SCA version of Ye-Net) [106] and the SCA-SRNet (which is the SCA version of SRNet) [10]. The idea is to use a network which is used for non-informed steganalysis and to inject not only the image to be steganalyzed, but also the modification probability map. It is thus assumed that Eve knows, or can have a good estimation [85] of the modification probability map, i.e. Eve has access to side-channel information.

The modification probability map is given to the pre-processing block SCA-Ye-Net [106], and equivalently to the first convolution block for SCA-SRNet [10], but the kernel values are replaced by their absolute values. After the convolution, each feature map is summed point-wise with the corresponding convolved "modification probability map" (see Figure 1.12). Note that the activation functions of the first convolutions in SCA-Ye-Net, i.e. the truncation activation function (*truncated linear unit (TLU)* in the article), are replaced by a ReLU. This makes it possible to propagate (forward pass) "virtually" throughout the network, an information related to the image, and another related to the modification probability map.

Note that this procedure to transform a Not-SCA-CNN into an SCA-CNN is inspired by the propagation of the modification probability map proposed in [22] and [21]. These two papers come as an improvement on the previous maxSRM Rich Models [20]. In maxSRM, instead of accumulating the number of occurrences in the co-occurrence matrix, an accumulation of the maximum of a local probability was used. In [22] and [21], the idea was to transform the modification probability map in a similar way to the filtering of the image, and then to update the co-occurrence matrix using the transformed version of the modification probability map, instead of the original modification probability map. The imitation of this principle was initially integrated into Ye-Net for CNN steganalysis, and this concept is easily transposable to most of the

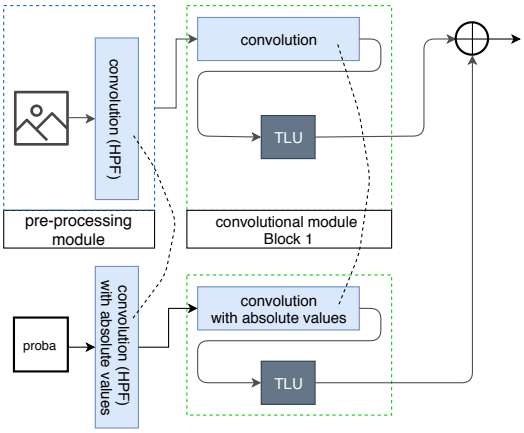

Figure 1.12 Integration of the modification probability map in a CNN.

modern CNNs.

1.5.3 THE JPEG STEGANALYSIS

The best JPEG CNN at the end of 2018 was SRNet [10]. Note that this network, at this period, is the only one that has been proposed with a Side Channel Aware (SCA) version.

It is interesting to list and rapidly discuss the previous CNNs used for JPEG steganalysis. The first network, published in February 2017, was the Zeng *et al.* network and was evaluated with a million images, and does a limited evaluation of stego-mismatch [114] [113]. Then in June 2017 at IH&MMSec'2017, two networks have been proposed: PNet [14], and Xu-Net-Jpeg [102]. Finally, SRNet [10] was added online in Septembre 2018.

In Zeng et al.s' network [114] [113], the pre-processing block takes as input a de-quantized (real value) image, then convolved it with 25 DCT basis, and then quantized and truncated the 25 filtered images. This pre-processing block, uses handcrafted filter kernels (DCT basis), the kernels' values are fixed, and these filters are inspired by DCTR Rich Models [41]. There are three different quantizations, so, the pre-processing block gives 3×25 residual images. The CNN is then made of 3 sub-networks which are each producing a feature vector of 512 dimension. The sub-networks are inspired by Xu-Net [101]. The three feature vectors, outputed by the three sub-networks, are then given to a fully connected structure, and the final network ends with a softmax layer.

Similarly to what has been done for spatial steganalysis, this network is using a pre-processing block inspired by Rich Models [41]. Note that the most efficient Rich Models today is the Gabor Filter Rich Models [100]. Also, note

that this network takes advantage of the notion of an ensemble of features, which comes from the three different sub-networks. The network of Zeng et al. is less efficient than Xu-Net-Jpeg [102], but gives an interesting first approach guided by Rich Models.

The PNet main idea (and also VNet which is less efficient but takes less memory) [14] is to imitate Phase-Aware Rich Models, such as DCTR [41], PHARM [42], or GFR [100], and therefore to have a decomposition of an input image into 64 features maps which represents the 64 phases of the Jpeg images. The pre-processing block takes as input a de-quantized (real value) image, convolves it with four filters, the "SQUARE5×5" from the Spatial Rich Models [28], a "point" high-pass filter (referenced as "catalyst kernel") which complements the "SQUARE5×5", and two directional Gabor Filters (angles 0 and $\pi/2$).

Just after the second block of convolution, a "PhaseSplit Module" splits the residual image into 64 feature maps (one map = one phase), similarly to what was done in Rich Models. Some interesting methods have been used such as (1) the succession of the fixed convolutions of the pre-processing block, and a second convolution with learnable values, (2) a clever update of BN parameters, (3) the use of the "Filter Group Option" which virtually builds sub-networks, (4) bagging on 5-cross-validation, (5) taking the 5 last evaluations in order to give the mean error for a network, (6) shuffling the database at the beginning of each epoch, to have better BN behavior, and to help generalization, and (7) eventually using an Ensemble. With such know-how, PNet beat the classical two-step machine learning approaches in a Not-SCA, and also in a SCA version (Ensemble Classifier + GFR).

The Xu-Net-Jpeg [102] is even more attractive since the approach was slightly better than PNet, and does not require a strong domain inspiration like in PNet. The Xu-Net-Jpeg is strongly inspired by ResNet [35], a well-established network from the machine learning community. ResNet allows the use of deeper networks thanks to the use of shortcuts. In Xu-Net-Jpeg, the pre-processing block takes as input a dequantized (real value) image, then convolves the image with 16 DCT basis (in the same spirit as Zeng et al. network [114] [113]), and then applies an absolute value, a truncation, and a set of convolutions, BN, ReLU until it obtains a feature vector of 384 dimension, which is given to a fully connected block. We can note that the max pooling or average pooling are replaced by convolutions. This network is really simple and was in 2017, the state-of-the-art method. In a way, these kind of results shows us that the networks proposed by machine learning community are very competitive and there is not so much domain-knowledge to integrate to the topology of a network in order to obtain a very efficient network.

In 2018 the state-of-the-art CNN for JPEG steganalysis (which can also be used for spatial steganalysis) was SRNet [10]. This network was previously presented in Section 1.5.1. Note that for the side channel aware version of SRNet, the *embedding change probability* per DCTs coefficient is, first, mapped

back in the spatial domain using absolute values for the DCT basis. This *side-channel map* then enters the network and is convolved with each kernel (this first convolution acts as a pre-processing block). Note that the convolutions in this first block for this *side-channel* map are such that the filter kernels are modified to their absolute values. After passing the convolution, the feature maps are summed with the square root of the values from the convolved *side-channel* map. Note that this idea is similar to what was exposed in SCA Ye-Net version (SCA-TLU-CNN) [106] about the integration of a Side-Channel map, and to the recent proposition for Side-Channel Aware steganalysis in JPEG with Rich Models [21], where the construction of the *side-channel* map, and especially the quantity $\delta_{uSA}^{1/2}$ [9] was defined.

Note that a similar solution with more convolutions, applied to the *side-channel map*, have been proposed in IH&MMSec'2019 [45].

1.5.4 DISCUSSION ABOUT THE MISMATCH PHENOMENON SCENARIO

Mismatch (cover-source mismatch or stego-mismatch) is a phenomenon present in machine learning, and this issue sees classification performances decrease because of the inconsistency between the distribution of the learning database and the distribution of the test database. The problem is not due to an inability to generalize in machine learning algorithms, but due to the lack of similar examples occurring in the training and test database. The problem of mismatch is an issue that goes well beyond the scope of steganalysis.

In steganalysis the phenomenon can be caused by many factors. The cover-source mismatch can be be caused by the use of different photo-sensors, by different digital processing, by different camera settings (focal length, ISO, lens, etc), by different image sizes, by different image resolutions, etc [30], [8]. The stego-mismatch can be caused by different amounts of embedded bits, or by different embedding algorithms.

Even if not yet fully explored and understood, the mismatch (cover-source mismatch (CSM) or stego mismatch) is a major area for examination in the coming years for the discipline. The results of the Alaska challenge [18] [10] published at the ACM conference IH&MMSec'2019 will continue these considerations.

In 2018, CSM had been established for 10 years [12]. There are two major current school of thought, as well as a third more exotic one:

- The first school of thought is the so-called **holistic** approach (that is to say, global, macroscopic, or systemic), and consists of learning all

[9] uSA stands for Upper bounded Sum of Absolute values.

[10] Alaska: A challenge of steganalysis into the wilderness of the real world. `https://alaska.utt.fr/`.

distributions [69], [68]. The use of a single CNN with millions of images [113] is in the logical continuation of this current school of thought. Note that this scenario does not consider that the test set can be used during learning. This scenario can be assimilated to an *online scenario* where the last player (from a game theory point of view) is the steganographer because in an online scenario the steganographer can change her strategy while the steganalyzer cannot.

- The second school of thought is **atomistic** (= partitioned, microscopic, analytical, of divide-and-conquer type, or individualized) and consists of partitioning the distribution [73], that is to say to create a partition and to associate a classifier for each cell of the partition. Note that an example of an atomistic approach for stego-mismatch management, using a CNN multi-classifier, is presented in [11] (a class is associated with each embedding algorithm - there is thus a latent partition). Note that this idea [11], among others, has been used by the winners of the Alaska challenge [110]. Note that again, this scenario does not consider that the test set can be used during learning. This scenario can also be assimilated to an *online scenario* where the last player (from a game theory point of view) is the steganographer because in an online scenario the steganographer can change her strategy while the steganalyst cannot.

- Finally, the third exotic school of thought considers that there is a test database (with much more than one image), and that the database is available, and usable (without labels) during learning. This scenario can be assimilated to an *offline scenario* where the last player (from a game theory point of view) is the steganalyser, because in this offline scenario the steganalyser is playing a more forensic role. In this situation, there are approaches of type domain adaptation, or a transfer of features GTCA [66], IMFA [55], CFT[24], where the idea is to define an invariant latent space. Another approach is ATS [61] which performs an unsupervised classification using only the test database and requires the embedding algorithm in order to re-embed a payload in the images from the test database.

These three schools of thought can help derive approaches by CNN that integrate the ideas presented here. That said, the ultimate solution may be to detect the phenomenon of mismatch and raise the alarm or prohibit the decision [50]. In short, to integrate a more intelligent mechanism than just holistic or atomistic.

1.6 STEGANOGRAPHY BY DEEP-LEARNING

In Simmons' founding article [90], steganography and steganalysis are defined as a *3-player game*. The steganographers, usually named Alice and Bob, want to exchange a message without being suspected by a third party. They must use a harmless medium, such as an image, and hide the message in this medium. The steganalyst, usually called Eve, observes the exchanges between Alice and Bob. Eve must check whether these images are natural, that is to say, cover images, or whether they hide a message, i.e. stego images.

This notion of *game* between Alice, Bob and Eve corresponds to that found in game theory. Each player tries to find a strategy that maximizes their chances of winning. For this, we express the problem as a min-max problem that we seek to optimize. The solution to the optimum, if it exists, is called the solution at the Nash equilibrium. When all the players are using a strategy at the Nash equilibrium, any change of strategy from a player, leads to a counter attack from the other players allowing them to increase their gains.

In 2012, Schöttle and Böhme [83], [84] have modeled with a simplifying hypotheses a problem of steganography and steganalysis and proposed a formal solution. Schöttle and Böhme have named this approach the *optimum adaptive steganography* or *strategic adaptive steganography* in opposition to the so-called *naive adaptive steganography* that corresponds to what is currently used in algorithms like HUGO (2010) [76], WOW (2012) [39], S-UNIWARD / J-UNIWARD / SI-UNIWARD (2013) [43], HILL (2014) [63], MiPOD (2016) [87], Synch-Hill (2015) [19], UED (2012) [32], IUERD (2016) [72], IUERD-*UpDist-Dejoin2* (2018) [65], etc.

That said, the mathematical formalization of the steganography / steganalysis problem by game theory is difficult and often far from practical in reality. Another way to determine a Nash equilibrium is to "simulate" the game. From a practical point of view, Alice plays the entire game alone, meaning that she does not interact with Bob or Eve to build her embedding algorithm. The idea is that she uses 3 algorithms (2 algorithms in the simplified version) that we name *agents*. Each of these agents will play the role of Alice, Bob [11] and Eve, and each agent runs at Alice's home. Let us note these three algorithms running at Alice's home: *Agent-Alice*, *Agent-Bob*, and *Agent-Eve*. With these notations, we thus make a distinction with the Human users: Alice (sender), Bob (receiver), and Eve (warden), and it allows us to highlight the fact that the three agents are executed from Alice's side. So, Agent-Alice's role is to embed a message into an image so that the resulting stego image is undetectable by Agent-Eve, and such that Agent-Bob can extract the message.

Alice can launch the game, that is to say the simulation, and the agents are "fighting" [12]. Once the agents have reached a Nash's equilibrium, Alice stops

[11] Bob is deleted in the simplified version.

[12] The reader should be aware that from a game theory point of view there are only two

the simulation and can now keep Agent-Alice, which is her *strategic adaptive embedding* algorithm, and can send Agent-Bob i.e the extraction algorithm (or any equivalent information) to Bob [13]. The secret communication between Alice and Bob is now possible through the use of the Agent-Alice algorithm for embedding and Agent-Bob algorithm for the extraction.

The first precursor approaches aimed at simulating a *strategic adaptive equilibrium*, and therefore proposing *strategic embedding* algorithms date from 2011 and 2012. The two approaches are MOD [25] and ASO [57] [56]; See Figure 1.13. Whether for MOD or ASO, the game is made by pitting Agent-Alice and Agent-Eve against each other. In this game, Agent-Bob is not used since Agent-Alice is simply generating a cost map, which is then used for coding and embedding the message thanks to an STC [26]. Alice can generate a cost map for a source image with the Agent-Alice, and then she can easily use the STC [26] algorithm to embed her message and obtain the stego image. From his side, Bob only has to use the STC [26] algorithm to retrieve the message from the stego image.

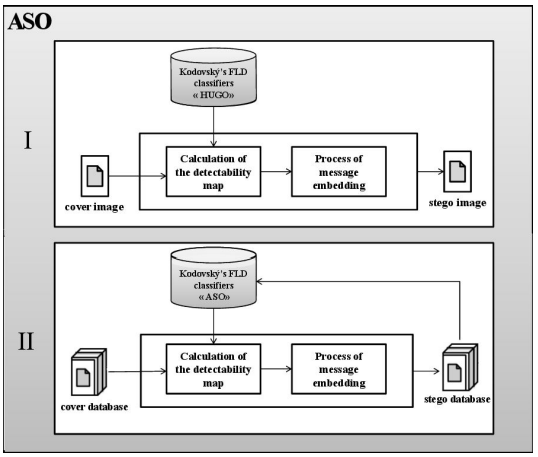

Figure 1.13 General scheme of ASO [57] [56].

In both MOD or ASO, the "simulation" is such that the two following actions are iterated until a stop criterion is reached:

teams that are competing (Agent-Alice plus Agent-Bob from one side, and Agent-Eve from the other side) in a zero-sum game.

[13] Note that the exchange of any secret information between Alice and Bob, prior to the use of Agent-Alice and Agent-Bob, requires the use of another steganographic channel. Also note that this initial sending from Alice to Bob before been able to use Agent-Alice and Agent-Bob is equivalent to the classical stego-key exchange problem.

i) Agent-Alice updates its embedding cost map by asking an Oracle (the Agent-Eve) how best to update each embedding cost, to be even less detectable.

In MOD (2011) [25], Agent-Eve is an SVM. Agent-Alice updates their embedding costs by reducing the SVM margin separating the covers and the stegos.

In ASO (2012) [57], Agent-Eve is an Ensemble Classifier [51] and is named an Oracle. Agent-Alice updates their embedding costs by transforming a stego in a cover.

In both cases, the idea is to find a displacement in the latent space (feature space) co-linear to the orthogonal axis to the hyperplane separating the cover and stego class. Note that in the current terminology, introduced by Ian Goodfellow in 2014 [31], Agent-Alice runs an adversarial attack, and the Oracle (Agent-Eve), named a discriminator (or the classifier to be deceived), must learn to counter this attack.

ii) The Oracle (Agent-Eve) updates its classifier. Reformulated with the terminology from machine learning, this equates to the discriminant update by re-learning it, in order to steganalysis once more the stego images generated by Agent-Alice.

In 2014, Goodfellow *et al.* [31] used neural networks to "simulate" a game with an *image generator network* and a *discriminating network* whose role was to decide whether an image was real or synthesized. The authors have named this Generative Adversarial Networks (GAN approach). The terminology used in this paper was subsequently widely adopted. Moreover, the use of neuron networks makes the expression of the min-max problem easy. The optimization is then carried out via the back-propagation optimization process. Moreover, thanks to deep-learning libraries it is now easy to build a GAN type system. As we have already mentioned before, the concept of game simulation, existed in steganography / steganalysis with MOD [25] and ASO [57], but the implementation and the optimization becomes easier with neural networks.

From 2017, after a period of 5 years of stagnation, the concept of the simulated game is once again studied in the field of steganography / steganalysis, thanks to the emergence of deep learning and GAN approaches. At the end of 2018, we can define four groups or four families [14] of approaches; some of which will probably merge:

[14] "Deep Learning in Steganography and Steganalysis since 2015", tutorial given at the "Image Signal & Security Mini-Workshop", the 30th of October 2018, IRISA / Inria Rennes, France, DOI: 10.13140/RG.2.2.25683.22567, *http://www.lirmm.fr/ chaumont/publications*. See the slides here, and the video of the talk here.

- The family by synthesis,
- The family by generation of the modifications probability map,
- The family by adversarial-embedding *iterated* (approaches misleading a discriminant),
- The family by 3-player game,

1.6.1 THE FAMILY BY SYNTHESIS

The first approaches based on *image synthesis* via a GAN [31] generator proposed the generation of cover images and then use them to make insertion by modification. These early propositions were approaches *by modification*. The argument put forward for such approaches is that the generated database would be safer. A reference often cited is that of SGAN [98] found on ArXiv, which was rejected at ICLR'2017 and was subsequently never published. This unpublished paper has a lot of errors and lack of proof. We should rather prefer the reference of SSGAN [89] that was published in September 2017, and that proposes the same thing: generate images and then hide messages in them. However, this protocol seems to complicate the matter. It is more logical that Alice herself chooses natural images that are safe for embedding, i.e. images that are innocuous, never broadcasted before, adapted to the context, with lots of noise or textures [88], not well classified by a classifier [56] or with a small deflection coefficient [87], rather than generating images, and then using them to hide a message.

A much more interesting approach using *synthesis* is to directly generate images that will be considered stego. To my knowledge, the first approach exploiting the GAN mechanism for image synthesis using the principle of steganography *without modifications* [27] is proposed in the article of Hu *et al.* [44] and published in July 2018; See Figure 1.14.

The first step consists of deriving a network able to synthesize images. In this paper, the DCGAN generator [82] is used to synthesize images with a preliminary learning thanks to GAN methodology. When fed with a vector of a fixed-size uniformly distributed in $[-1, 1]$ the generator synthesizes an image. The second step consists of learning to another network to extract a vector from a synthesized image; the extracted vector must correspond to the vector given at the input of the generator which synthesizes the image. Finally, the last step consists of sending Bob the extraction network. Now, Alice can map a message to a fixed-size uniformly distributed vector, and then synthesize an image with the given vector, and send it to Bob. Bob can extract the vector and retrieve the corresponding message.

The approaches with *no modifications* have been around for many years, and it is known that one of the problems is that the number of bits that can be communicated is lower compared to the approaches with modifications. That

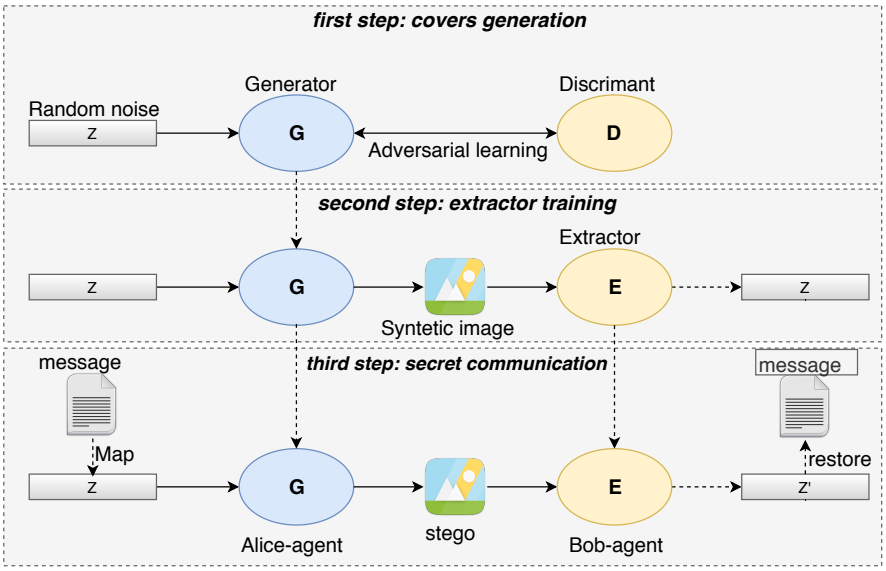

Figure 1.14 Hu *et al.* [44] approach by synthesis without modification.

said, the gap between the approaches by *modifications* versus *no-modifications* is beginning to narrow.

Here is a rapid analysis of the efficiency of the method. In the paper of Hu *et al.* [44], the capacity is around 0.018 bits per pixel (bpp) with images 64×64 pixels [15]. In the experiment carried out, the synthesized images are either faces or photographs of food. An algorithm like HILL[63] (one of the most powerful algorithms on the BOSS database [88]) is detected by SRNet [10] (one of the most successful steganalysis approaches towards the end of 2018) with a probability of error of Pe $= 31.3\%$ (note that a Pe of 50% is equivalent to a random detector) on a 256×256 pixels BOSS database, for a payload size of 0.1 bpp. Due to the square root law, the Pe would be higher for the 64×64 pixels BOSS database.

Therefore, there is around 0.02 bpp for the unmodified synthetic approach of Hu *et al.* [44] whose security has not yet been evaluated enough, against something around 0.1 bpp for HILL, with less than one chance in three to be detected with a *clairvoyant* steganalysis i.e. a laboratory steganalysis (un-

[15] The vector dimension is 100. This vector is used to synthesize images of a size $64 \times 64 \times 3$. There are 100×3 bits (see the mapping) per image, i.e. about 0.02 bits per pixel (bpp). The Bit Error Rate is BER $= 1 - 0.94 = 6\%$. It is, therefore, necessary to add an Error Correcting Code (ECC) so that the approach is without errors. With the use of a Hamming code $[15, 11, 3]$ that corrects at best 6% of errors, the payload size is therefore around 0.018 bpp.

realistic and much more efficient than a "real-world" / "into the wild" steganalysis [48] [18]). Therefore, there is still a margin in terms of the number of bits transmitted between the *no-modification* synthesis-based approaches, such as that of Hu *et al.* approach [44], and *modification* approaches such as S-UNIWARD [43], HILL [63], MiPod [87] or even Synch-Hill [19], but this margin has been reduced [16]. Also, note that there are still some issues to be addressed to ensure that approaches such as the one proposed by Hu *et al.* are entirely safe. In particular, it must be ensured that the detection of synthetic images [81] does not compromise the communication channel in the long term. It must also be ensured that the absence of a secret key does not jeopardize the approach. Indeed, if one considers that the generator is public, is it possible to use this information to deduce that a synthesis approach without modification has been used?

1.6.2 THE FAMILY BY GENERATION OF THE MODIFICATIONS PROBABILITY MAP

The family by generation of the modification probability map is summarized in late 2018 in two papers: ASDL-GAN [95], and UT-6HPF-GAN [104]; See Figure 1.15. In this approach, there is a generator network and a discriminant network. From a cover, the generator network generates a map which is named the modification probability map. This modification probability map is then passed to an equivalent of the random draw function used in the STC [26] simulator. We then obtain a map whose values belong to {-1, 0, +1}. This map is called the modification map and corresponds to the so-called stego-noise. The discriminant network takes as input a cover or an image resulting from the summation (point-to-point sum) of the cover and the stego-noise generated by the generator. The discriminant's objective is to distinguish between the cover and the "*cover + stego-noise*" image. The generator's objective is to generate a modification map which makes it possible to mislead the discriminant the most. Of course, the generator is forced to generate a non-zero probability map by adding in the loss term, a term constraining the size of the payload in addition to the term misleading the discriminant.

In practice, taking the latest approach UT-6HPF-GAN [104], the generator is a U-Net type network, the draw function is obtained by a differentiable function *double Tanh*, and the discriminant is the Xu-Net [101] enriched with 6 high-pass filters for the pre-processing in the same spirit as Ye-Net [106] or Yedroudj-Net [108].

The system learns on a first database, and then security comparisons are made on the 256×256 pixels BOSS [4], LIRMMBase [78], and BOWS2 [3]

[16] The other families of steganography by deep learning, which are *modification* based, will probably help to maintain this performance gap for a few years more.

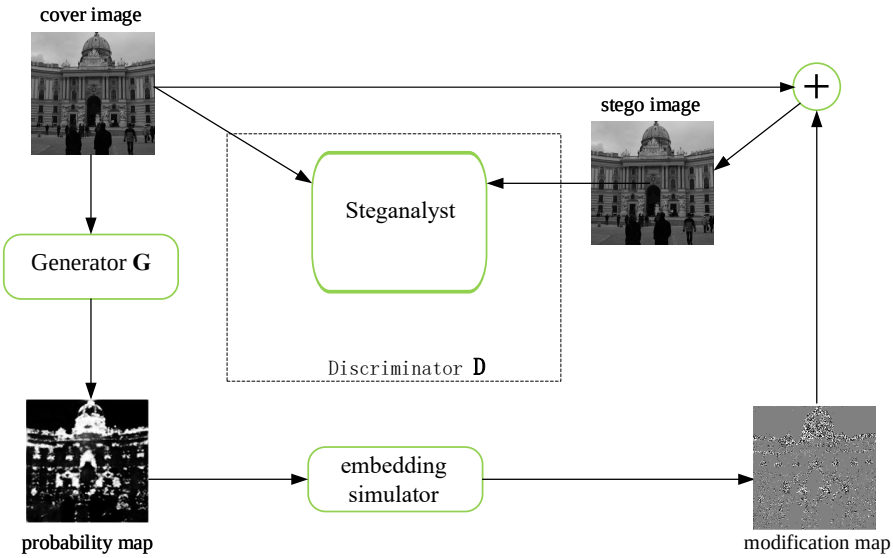

Figure 1.15 ASDL approach; generation of the modifications probability map

databases. The steganalysis is done with the Ensemble Classifier (EC) [51] fed by SRM [28], with EC plus the MaxSRM [20], and with Xu-Net [101]. Note that using Xu-Net is not a good choice since it is less efficient than EC+SRM or EC+MaxSRM, and also because it is the discriminant in the UT-6HPF-GAN (there is a risk of falling into an "incompleteness" issue; see papers [52] [54]). So, only looking at the results with EC+SRM, on the BOSS database, with real embedding using STC [26], the performances are equivalent to those of HILL [63], which is one of the most efficient embedding algorithms on BOSS [88]. It is therefore a very promising family.

Additionally, the generator does not seem to be impacted when used on a database that is different from the learning database. Nevertheless, curriculum learning has to be used when the target payload is changed, which seems to indicate a kind of sensitivity to the mismatch. Further reflexions have also to be achieved related to the generator's loss, and the mixing of both a security-related term and a payload-size term. Usually, one of the two criteria is fixed, so that we have to be in a payload-limited sender scenario or a security-limited sender scenario. Note that a version for JPEG has been proposed in IH&MMSec'2019, JS-GAN [105].

1.6.3 THE FAMILY BY ADVERSARIAL-EMBEDDING *ITERATED* (APPROACHES MISLEADING A DISCRIMINANT)

The family by adversarial-embedding iterated re-uses the concept of *game simulation* which was presented in the beginning of Section 1.6 with a simplification of the problem since there are only two-players: Agent-Alice and Agent-Eve. Historically MOD [25] and ASO [57] were the first algorithms of this type.

Recently some papers have used the adversarial concept[17] by generating a deceiving example (see [116]), but these approaches are not adversarial-embedding iterated. Nor are they dynamic, they contain no game simulation, they are not trying to reach a Nash equilibrium, there is no learning alternation between the embedder and the steganalysis.

A paper whose spirit is more in tune with a simulation of a game, which takes the principle of ASO [57], and whose objective is to update the cost map is the algorithm ADV-EMB [96] (previously named AMA on *ArXiv* arXiv:1803.09043). In this article, the authors propose to make an adversarial-embedding *iterated*, by letting Agent-Alice access the gradient of the loss of Agent-Eve (similarly to ASO, where Agent-Alice has access to its Oracle (the Agent-Eve)). In ADV-EMB, Agent-Alice uses the gradient, of the direction to the class frontier (between classes cover and stego), to modify the cost map, and in ASO, Agent-Alice directly uses the direction of the class frontier to modify the cost map.

In ADV-EMB [96], the cost map is initialized with the cost of in S-UNIWARD (for ASO it was the cost of HUGO [76]). During the iterations, the cost map is updated, but there is only a β percentage of values that are updated [18]. When the ADV-EMB iterations are stopped, the cost map is composed of a $\beta - 1$ percent of positions having a cost defined by S-UNIWARD, and β percent of positions having a cost coming from a change in the initial cost given by S-UNIWARD.

Note that updating a cost causes a cost asymmetry since the cost of a +1 change is no longer equal to the cost of a -1 change, as in ASO. Besides, the update of the two costs of a pixel is rather rough since it is a simple division by 2 for a direction (+1 or -1) and multiplication by 2 for the other direction. The sign of the gradient of loss, calculated by choosing the cover label, for a given pixel, makes it possible to determine for each of the two directions (+ 1 / -1) if we should reduce or increase the cost. The idea is as in ASO, to

[17] An adversarial attack does not necessarily require us to use a deep learning classifier.

[18] In STC, before coding the message, the pixels position of the image are shuffled thanks to the use of a pseudo-random shuffler, seeded by the secret stego-key. Note that this stego-key is shared between Alice and Bob. After the shuffling step, ADV-EMB selects the last β percent pixels of the *shuffled* image, and modifies their associated cost and only those ones.

deceive the discriminant since when we decide to reduce the value of a cost, it is to favor the direction of modification associated with this cost, and thus we promote getting closer to the cover class.

With such a scheme, security is improved. The fact that it is preferable to have a small number of modifications to the initial cost map probably makes it possible to preserve the initial embedding approach, and therefore not to introduce too many traces that could be detected by another steganalyzer [54]. That said, the update to the costs should probably be refined to better take into account the value of the gradient. For the moment, the selection of the β percent of pixels that will be modified is suboptimal, and this selection should eventually be done by looking at the initial cost of the whole pixel. Finally, as it is the case for ASO, if the discriminant is not powerful enough to carry out a steganalysis, then it can be totally counterproductive for the Agent-Alice. Therefore, there are many open questions regarding the convergence criterion, the stopping criterion, the number of iterations in the alternation between Agent-Alice and Agent-Eve, and the definition of a metric for measuring the relevance of Agent-Eve, etc. Note that an adversarial embedding *iterated* with Agent-Alice countering multiple versions of Agent-Eve has been proposed in IH&MMSec'2019 [7].

1.6.4 THE FAMILY BY 3-PLAYER GAME

The 3-player game concept is an extension of the previous family (see the family "adversarial-embedding *iterated*"), but this time with three agents and all are neural networks. Here, the three agents: Agent-Alice, Agent-Bob, and Agent-Eve are present (see Section 1.6 for an overview of the game). Note that Agent-Alice and Agent-Bob are "linked" since Agent-Bob is only there to add a constraint on the solution obtained by Agent-Alice. Thus, the primary "game" is an antagonistic (or adversarial) game between Agent-Alice and Agent-Eve, while the "game" between Agent-Alice and Agent-Bob is rather cooperative, since these two agents share the common purpose of communicating (Agent-Alice and Agent-Bob both want Agent-Bob to be able to extract the message without errors). Figure 1.16, from [109] summarizes the principle of the 3-player game. Agent-Alice takes a cover image, a message and a stego-key, and after a discretisation step generates a stego image. This stego image is used by Agent-Bob to retrieve a message. On the other side, Agent-Eve has to decide whether an image is cover or stego; this agent outputs a score.

Historically, after MOD and ASO, which only included two players, we can see the premise of the idea of three players appear in 2016 with the paper of Abadi and Andersen [1]. In this paper, Abadi and Andersen [1] from Google Brain, proposed a cryptographic toy-example for an encryption based on the use of three neural networks. The use of neural networks makes it easy to obtain a *strategic equilibrium* since the problem is expressed as a min-

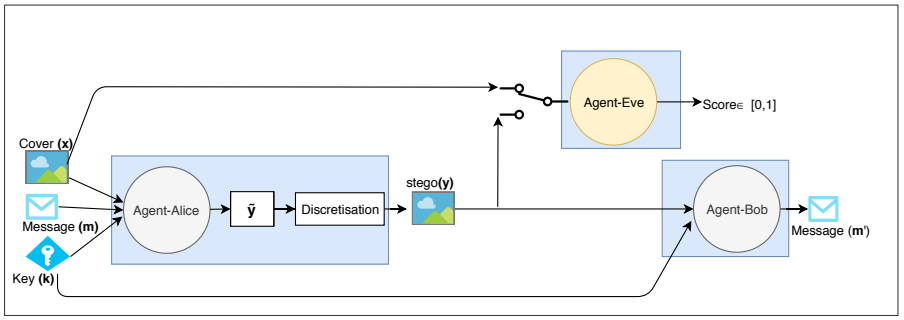

Figure 1.16 The overall architecture of the *3-players game*.

max problem and its optimization can be carried out by the back-propagation process. Naturally, this 3-player game concept can be transposed to steganography with the use of deep learning.

In December 2017 (GSIVAT; [33]), and in September 2018 (HiDDeN; [117]), two different teams from the machine learning community proposed, in NIPS'2017, then in ECCV'2018, to achieve *strategic embedding* thanks to 3 CNNs, iteratively updated, who play the role of Agent-Alice, Agent-Bob, and agent Agent-Eve. These two articles do not rigorously define the concept of the 3-player game, and there are erroneous assertions, mainly because the security and its evaluation are not correctly handled. If we place ourself in the standard framework to evaluate the empirical security of an embedding algorithm, that is to say with a clairvoyant Eve, the two approaches are very detectable. The most significant issues with these two papers are first, neither of the two approaches uses a stego-key; which is the equivalent to always using the same key, and it leads to very detectable schemes [78], second, there is no discretization of pixel values issued from Agent-Alice, third, the computational complexity, due to the use of fully connected blocks, leads to un-practical approaches, and fourthly, the security evaluation is not carried out with a state-of-the-art steganalyzer.

At the beginning of 2019, Yedroudj et al. [109] redefined the 3-player concept, by integrating the possibility of using a stego-key, treating the problem of discretization, going through convolution modules to have a scalable solution, and using a suitable steganalyzer. The proposition is not comparable to classical adaptive embedding approaches, but there is a real potential to such an approach. The Bit Error Rate is sufficiently small to be nullified, the embedding is done in the texture parts, and security could be improved in the future. As an example, the probability of error with a steganalysis by Yedroudj-Net[108], under equal errors prior, for a real payload size 0,3 bpp[19]

[19] A Hamming error correcting code ensures a null BER theoretically for most of the images,

for images of from BOWS2 database is 10.8%. This can, for example, be compared to the steganalysis of WOW[39] using the same conditions, which give a probability error of 22.4%. There is still a security gap, but this approach paves the way to much research. There are still open questions on the link between Agent-Alice and Agent-Bob, on the use of GANs, and on the definition of losses and the tuning of the compromises between the different constraints.

CONCLUSION

In this chapter, we have practically completed a full presentation of the subject on deep learning in steganography and steganalysis, since its appearance in 2015. As a reviewer of many papers related to this subject during the period 2015 - 2018, I think and I hope this chapter will help the community to understand what has been done and what are the next directions to explore.

In this chapter, we recalled the main elements of a CNN. We discussed the memory and time complexity, and practical problems for efficiency. We explored the link between some past approaches sharing similarities with what is currently carried out in a CNN. We presented the various main networks until the beginning of 2019, and multiple scenarios, finally we touched on the recent approaches for steganography with deep learning.

As mentioned in this chapter, many things have not been solved yet, and the major issue is to be able to experiment with more realistic hypotheses to be more "into the wild". The "holy grail" is cover-source mismatch and stego-mismatch, but in a way, the mismatch is a problem shared by the whole machine learning community. CNNs are now very present in the steganalysis community, and the next question is probably: how to go a step further and produce clever networks?

REFERENCE

and thus a rate of 0.3 bpp for these images.

[1] Martín Abadi and David G. Andersen. Learning to Protect Communications with Adversarial Neural Cryptography. In *Unpublished - ArXiv*, volume abs/1610.06918, 2016. URL `http://arxiv.org/abs/1610.06918`.

[2] Hasan Abdulrahman, Marc Chaumont, Philippe Montesinos, and Baptiste Magnier. Color Image Steganalysis Based On Steerable Gaussian Filters Bank. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2016, pages 109–114, Vigo, Galicia, Spain, June 2016.

[3] Patrick Bas and Teddy Furon. BOWS-2 Contest (Break Our Watermarking System), 2008. Organized between the 17th of July 2007 and the 17th of April 2008. http://bows2.ec-lille.fr/.

[4] Patrick Bas, Tomas Filler, and Tomas Pevný. 'Break Our Steganographic System': The Ins and Outs of Organizing BOSS. In *Proceedings of 13th International Conference on Information Hiding, IH'2011*, volume 6958 of *Lecture Notes in Computer Science, Springer*, pages 59–70, Prague, Czech Republic, May 2011.

[5] Belhassen Bayar and Matthew C. Stamm. A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2016, pages 5–10, Vigo, Galicia, Spain, June 2016.

[6] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives. *IEEE Transaction on Pattern Analysis and Machine Intelligence, PAMI*, 35(8):1798–1828, 2013.

[7] Solène Bernard, Tomás Pevný, Patrick Bas, and John Klein. Exploiting Adversarial Embeddings for Better Steganography. In *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2019, pages 216–221, Paris, France, July 2019.

[8] Dirk Borghys, Patrick Bas, and Helena Bruyninckx. Facing the Cover-Source Mismatch on JPHide using Training-Set Design. In *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec'2018*, pages 17–22, Innsbruck, Austria, June 2018.

[9] Mehdi Boroumand and Jessica Fridrich. Nonlinear Feature Normalization in Steganalysis. In *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2017, pages 45–54, Philadelphia, Pennsylvania, USA, June 2017.

[10] Mehdi Boroumand, Mo Chen, and Jessica Fridrich. Deep Residual Network for Steganalysis of Digital Images. *IEEE Transactions on Information Forensics and Security*, 14(5):1181 – 1193, May 2019.

[11] Jan Butora and Jessica J. Fridrich. Detection of Diversified Stego Sources with CNNs. In *Proceedings of Media Watermarking, Security, and Forensics, MWSF'2019, Part of IS&T International Symposium on Electronic Imaging, EI'2019*, pages 534(1)–534(11), Burlingame, California, USA, January 2019.

[12] Giacomo Cancelli, Gwenaël J. Doërr, Mauro Barni, and Ingemar J. Cox. A Comparative Study of +/-1 Steganalyzers. In *Proceedings of Workshop Multimedia Signal Processing, MMSP'2008*, pages 791–796, Cairns, Queensland, Australia, October 2008.

[13] M. Chaumont and S. Kouider. Steganalysis by Ensemble Classifiers with Boosting by Regression, and Post-Selection of Features. In *Proceedings of IEEE International Conference on Image Processing, ICIP'2012*, pages 1133–1136, Lake Buena Vista (suburb of Orlando), Florida, USA, September 2012.

[14] Mo Chen, Vahid Sedighi, Mehdi Boroumand, and Jessica Fridrich. JPEG-Phase-Aware Convolutional Neural Network for Steganalysis of JPEG Images. In *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2017, pages 75–84, Drexel University in Philadelphia, PA, June 2017.

[15] Mo Chen, Mehdi Boroumand, and Jessica J. Fridrich. Deep Learning Regressors for Quantitative Steganalysis. In *Proceedings of Media Watermarking, Security, and Forensics, MWSF'2018, Part of IS&T International Symposium on Electronic Imaging, EI'2018*, pages 160(1)–160(7), Burlingame, California, USA, 28 January - 2 February 2018.

[16] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR'2017*, pages 1800–1807, Honolulu, HI, USA, July 2017.

[17] Rémi Cogranne, Tomas Denemark, and Jessica Fridrich. Theoretical Model of the FLD Ensemble Classifier Based on Hypothesis Testing Theory. In *Proceedings of IEEE International Workshop on Information Forensics and Security, WIFS'2014*, pages 167–172, Atlanta, GA, December 2014.

[18] Rémi Cogranne, Quentin Giboulot, and Patrick Bas. The ALASKA Steganalysis Challenge: A First Step Towards Steganalysis. In *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2019, pages 125–137, Paris, France, July 2019.

[19] Tomas Denemark and Jessica Fridrich. Improving Steganographic Security by Synchronizing the Selection Channel. In *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2015, pages 5–14, Portland, Oregon, USA, 2015.

[20] Tomas Denemark, Vahid Sedighi, Vojetech Holub, Rémi Cogranne, and Jessica Fridrich. Selection-Channel-Aware Rich Model for Steganalysis of Digital Images. In *Proceedings of IEEE International Workshop on Information Forensics and Security, WIFS'2014*, pages 48–53, Atlanta, Georgia, USA, December 2014.

[21] Tomas Denemark, Mehdi Boroumand, and Jessica Fridrich. Steganalysis Features for Content-Adaptive JPEG Steganography. *IEEE Transactions on Information Forensics and Security*, 11(8):1736–1746, August 2016.

[22] Tomás Denemark, Jessica J. Fridrich, and Pedro Comesaña Alfaro. Improving Selection-Channel-Aware Steganalysis Features. In *Proceedings of Media Watermarking, Security, and Forensics, MWSF'2018, Part of IS&T International Symposium on Electronic Imaging, EI'2016*, pages 1–8, San Francisco, California, USA, February 2016.

[23] Xiaoqing Deng, Bolin Chen, Weiqi Luo, and Da Luo. Fast and Effective Global Covariance Pooling Network for Image Steganalysis. In *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2019, pages 230–234, Paris, France, July 2019.

[24] Chaoyu Feng, Xiang Wei Kong, Ming Li, Yong Yang, and Yanqing Guo. Contribution-Based Feature Transfer for JPEG Mismatched Steganalysis. In *Proceedings of IEEE International Conference on Image Processing, ICIP'2017*, pages 500–504, Beijing, China, September 2017.

[25] Tomas Filler and Jessica Fridrich. Design of Adaptive Steganographic Schemes for Digital Images. In *Proceedings of SPIE Media Watermarking, Security, and Forensics, Part of IS&T/SPIE 21th Annual Symposium on Electronic Imaging, SPIE'2011*, vol-

ume 7880, pages 78800F–78800F–14, San Francisco Airport, California, United States, February 2011.

[26] Tomas Filler, Jan Judas, and Jessica Fridrich. Minimizing Additive Distortion in Steganography Using Syndrome-Trellis Codes. *IEEE Transactions on Information Forensics and Security*, 6(3):920–935, September 2011.

[27] Jessica Fridrich. *Steganography in Digital Media*. Cambridge University Press, 2009. ISBN 9781139192903. Cambridge Books Online.

[28] Jessica Fridrich and Jan Kodovsky. Rich Models for Steganalysis of Digital Images. *IEEE Transactions on Information Forensics and Security, TIFS*, 7(3):868–882, June 2012.

[29] Jessica Fridrich, Jan Kodovský, Vojetech Holub, and Miroslav Goljan. Breaking HUGO - The Process Discovery. In *Proceedings of Information Hiding, 13th International Conference, IH'2011*, volume 6958 of *Lecture Notes in Computer Science, Springer*, pages 85–101, Prague, Czech Republic, May 2011.

[30] Quentin Giboulo to, Rémi Cogranne, and Patrick Bas. Steganalysis into the Wild: How to Define a Source? In *Proceedings of Media Watermarking, Security, and Forensics, MWSF'2018, Part of IS&T International Symposium on Electronic Imaging, EI'2018*, Burlingame, California, USA, 28 January - 2 February 2018.

[31] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Proceedings of Advances in Neural Information Processing Systems, NIPS'2014*, pages 2672–2680, December 2014.

[32] Linjie Guo, Jiangqun Ni, and Yun Qing Shi. An Efficient JPEG Steganographic Scheme Using Uniform Embedding. In *Proceedings of IEEE International Workshop on Information Forensics and Security, WIFS'2012*, pages 169–174, Costa Adeje, Tenerife, Spain, December 2012.

[33] Jamie Hayes and George Danezis. Generating Steganographic Images Via Adversarial Training. In *Proceedings of Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, NIPS'2017*, pages 1951–1960, Long Beach, CA, USA, December 2017.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In *Proceedings of the European Conference on Computer Vision, ECCV'2014*, pages 346–361, Zurich, Switzerland, September 2014.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR'2016*, pages 770–778, Las Vegas, Nevada, June 2016.

[36] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep Learning Scaling is Predictable, Empirically. In *Unpublished - ArXiv*, volume abs/1712.00409, 2017. URL `http://arxiv.org/abs/1712.00409`.

[37] Geoffrey. E. Hinton and Ruslan. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006.

[38] Vojtech Holub and Jessica Fridrich. Optimizing Pixel Predictors for Steganalysis. In *Proceedings of SPIE Media Watermarking, Security, and Forensics, Part of IS&T/SPIE 22th Annual Symposium on Electronic Imaging, SPIE'2012*, volume 8303, pages 830309–830309–13, San Francisco, California, USA, February 2012.

[39] Vojtech Holub and Jessica Fridrich. Designing Steganographic Distortion Using Directional Filters. In *Proceedings of the IEEE International Workshop on Information Forensics and Security, WIFS'2012*, pages 234–239, Tenerife, Spain, December 2012.

[40] Vojtech Holub and Jessica Fridrich. Random Projections of Residuals for Digital Image Steganalysis. *IEEE Transactions on Information Forensics and Security, TIFS*, 8(12): 1996–2006, December 2013.

[41] Vojtech Holub and Jessica Fridrich. Low-Complexity Features for JPEG Steganalysis Using Undecimated DCT. *IEEE Transactions on Information Forensics and Security, TIFS*, 10(2):219–228, February 2015.

[42] Vojtech Holub and Jessica Fridrich. Phase-Aware Projection Model for Steganalysis of JPEG Images. In *Proceedings of SPIE Media Watermarking, Security, and Forensics 2015, Part of IS&T/SPIE Annual Symposium on Electronic Imaging, SPIE'2015*, volume 9409, page 11, San Francisco, California, USA, February 2015.

[43] Vojtech Holub, Jessica Fridrich, and Tomas Denemark. Universal Distortion Function for Steganography in an Arbitrary Domain. *EURASIP Journal on Information Security, JIS*, 2014(1):1, 2014.

[44] Donghui Hu, Liang Wang, Wenjie Jiang, Shuli Zheng, and Bin Li. A Novel Image Steganography Method via Deep Convolutional Generative Adversarial Networks. *IEEE Access*, 6:38303–38314, July 2018.

[45] Junwen Huang, Jiangqun Ni, Linhong Wan, and Jingwen Yan. A Customized Convolutional Neural Network with Low Model Complexity for JPEG Steganalysis. In *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2019, pages 198–203, Paris, France, July 2019.

[46] Xiaosa Huang, Shilin Wang, Tanfeng Sun, Gongshen Liu, and Xiang Lin. Steganalysis of Adaptive JPEG Steganography based on ResDet. In *Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA'2018*, volume 2018, pages 12–15, Hononulu Hawaii, November 2018.

[47] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'2015, pages 448–456, Lille, France, July 2015.

[48] Andrew. D. Ker, Patrick Bas, Rainer Böhme, Rémi Cogranne, Scott Craver, Tomas Filler, Jessica Fridrich, and Tomas Pevný. Moving Steganography and Steganalysis from the Laboratory into the Real World. In *Proceedings of the 1st ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec'2013*, pages 45–58, Montpellier, France, June 2013.

[49] Diederik P. Kingma and Jimmy Lei Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of Conference on Learning Representations, ICLR'2015*, page 13, San Diego, CA, May 2015.

[50] Mustafa Anil Koak, David Ramirez, Elza Erkip, and Dennis Shasha. SafePredict: A Meta-Algorithm for Machine Learning That Uses Refusals to Guarantee Correctness. In *ArXiv*, volume abs/1708.06425, 2017. URL `http://arxiv.org/abs/1708.06425`.

[51] J. Kodovský, J. Fridrich, and Vojtech Holub. Ensemble Classifiers for Steganalysis of Digital Media. *IEEE Transactions on Information Forensics and Security*, 7(2): 432–444, 2012.

[52] Jan Kodovský and Jessica Fridrich. On Completeness of Feature Spaces in Blind Steganalysis. In *Proceedings of the 10th ACM Workshop on Multimedia and Security*,

MM&Sec'2008, pages 123–132, Oxford, United Kingdom, 2008.

[53] Jan Kodovský and Jessica J. Fridrich. Quantitative Steganalysis Using Rich Models. In *Proceeding of SPIE Media Watermarking, Security, and Forensics, Part of IS&T/SPIE 23th Annual Symposium on Electronic Imaging, SPIE'2013*, volume 8665 of *SPIE Proceedings*, page 111, San Francisco, California, USA, February 2013.

[54] Jan Kodovsky, Jessica Fridrich, and Vojtech Holub. On Dangers of Overtraining Steganography to Incomplete Cover Model. In *Proceedings of the Thirteenth ACM Multimedia Workshop on Multimedia and Security*, MM&Sec'2011, pages 69–76, Buffalo, New York, USA, September 2011.

[55] Xiangwei Kong, Chaoyu Feng, Ming Li, and Yanqing Guo. Iterative Multi-order Feature Alignment for JPEG Mismatched Steganalysis. *Journal of Neurocomputing*, 214(C):458–470, November 2016.

[56] Sarra Kouider, Marc Chaumont, and William Puech. Technical Points About Adaptive Steganography by Oracle (ASO). In *Proceeding of Signal Processing Conference, EUSIPCO'2012, 2012 Proceedings of the 20th European*, pages 1703–1707, Bucharest, Romania, August 2012.

[57] Sarra Kouider, Marc Chaumont, and William Puech. Adaptive Steganography by Oracle (ASO). In *Proceeding of the IEEE International Conference on Multimedia and Expo, ICME'2013*, pages 1–6, San Jose, California, USA, July 2013.

[58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceeding of Advances in Neural Information Processing Systems 25, NIPS'2012*, pages 1097–1105. Curran Associates, Inc., Lake Tahoe, Nevada, USA, December 2012.

[59] Artur Kuzin, Artur Fattakhov, Ilya Kibardin, Vladimir Iglovikov, and Ruslan Dautov. Camera Model Identification Using Convolutional Neural Networks. In *Proceedings of the 2nd International Workshop on Big Data Analytic for Cyber Crime Investigation and Prevention, co-located with IEEE Big Data 2018*, pages 3107–3110, Seattle, USA, December 2018.

[60] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, May 2015.

[61] Daniel Lerch-Hostalot and David Megías. Unsupervised Steganalysis Based on Artificial Training Sets. *Engineering Applications of Artificial Intelligence*, 50(C):45–59, April 2016.

[62] B. Li, W. Wei, A. Ferreira, and S. Tan. ReST-Net: Diverse Activation Modules and Parallel Subnets-Based CNN for Spatial Image Steganalysis. *IEEE Signal Processing Letters*, 25(5):650–654, May 2018.

[63] Bin Li, Ming Wang, Jiwu Huang, and Xiaolong Li. A New Cost Function for Spatial Image Steganography. In *Proceedings of IEEE International Conference on Image Processing, ICIP'2014*, pages 4206–4210, Paris, France, October 2014.

[64] Bin Li, Ming Wang, Xiaolong Li, Shunquan Tan, and Jiwu Huang. A Strategy of Clustering Modification Directions in Spatial Image Steganography. *IEEE Transaction on Information Forensics and Security*, 10(9):1905–1917, 2015.

[65] Weixiang Li, Weiming Zhang, Kejiang Chen, Wenbo Zhou, and Nenghai Yu. Defining Joint Distortion for JPEG Steganography. In *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2018, pages 5–16, Innsbruck, Austria, 2018.

[66] X. Li, X. Kong, B. Wang, Y. Guo, and X. You. Generalized Transfer Component

Analysis for Mismatched JPEG Steganalysis. In *Proceedings of IEEE International Conference on Image Processing, ICIP'2013*, pages 4432–4436, Melbourne, Australia, September 2013.

[67] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. Progressive Neural Architecture Search. In *Proceedings of the European Conference on Computer Vision, ECCV'2018*, volume 11205 of *Lecture Notes in Computer Science, Springer*, pages 19–35, Munich, Germany, September 2018.

[68] Ivans Lubenko and Andrew D. Ker. Going From Small to Large Data in Steganalysis. In *Proceedings of Media Watermarking, Security, and Forensics III, Part of IS&T/SPIE 22th Annual Symposium on Electronic Imaging, SPIE'2012*, volume 8303, San Francisco, California, USA, February 2012.

[69] Ivans Lubenko and Andrew D. Ker. Steganalysis with Mismatched Covers: Do Simple Classifiers Help? In *Proceedings of the 14th ACM Multimedia and Security Workshop, MM&Sec'2008*, MM&Sec'2012, pages 11–18, Coventry, United Kingdom, September 2012.

[70] Stéphane Mallat. Understanding Deep Convolutional Networks. *Philosophical Transactions of the Royal Society. Series A, Mathematical, physical, and engineering sciences*, 374, 2016.

[71] Michael A. Alcorn and Qi Li and Zhitao Gong and Chengfei Wang and Long Mai and Wei-Shinn Ku and Anh Nguyen. Strike (with) a Pose: Neural Networks Are Easily Fooled by Strange Poses of Familiar Objects. In *Proceedings of the International Conference on Computer Vision*, page 36, Seoul, Korea, October 2019.

[72] Yuanfeng Pan, Jiangqun Ni, and Wenkang Su. Improved Uniform Embedding for Efficient JPEG Steganography. In *Proceedings of the International Conference on Cloud Computing and Security, ICCCS(2016*, volume 10039 of *Part of the Lecture Notes in Computer Science book series (LNCS), Springer*, pages 125–133, Nanjing, China, July 2016.

[73] Jérôme Pasquet, Sandra Bringay, and Marc Chaumont. Steganalysis with Cover-Source Mismatch and a Small Learning Database. In *Proceedings of the 22nd European Signal Processing Conference, EUSIPCO'2014*, pages 2425–2429, Lisbon, Portugal, September 2014.

[74] Tomas Pevný. Co-occurrence Steganalysis in High Dimensions. In *Proceeding of SPIE Media Watermarking, Security, and Forensics, Part of IS&T/SPIE 22th Annual Symposium on Electronic Imaging, SPIE'2012*, volume 8303, pages 83030B–83030B–13, San Francisco, California, USA, February 2012.

[75] Tomas Pevný and Andrew D. Ker. The Challenges of Rich Features in Universal Steganalysis. In *Proceeding of SPIE Media Watermarking, Security, and Forensics, Part of IS&T/SPIE 23th Annual Symposium on Electronic Imaging, SPIE'2013*, volume 8665, pages 86650M–86650M–15, San Francisco, California, USA, February 2013.

[76] Tomas Pevný, Tomas Filler, and Patrick Bas. Using High-Dimensional Image Models to Perform Highly Undetectable Steganography. In *Proceedings of the 12th International Conference on Information Hiding, IH'2010*, volume 6387 of *Lecture Notes in Computer Science, Springer*, pages 161–177, Calgary, Alberta, Canada, June 2010.

[77] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient Neural Architecture Search via Parameters Sharing. In *Proceedings of Thirty-fifth International Conference on Machine Learning, ICML'2018*, page 11, Stockholm, Sweden,

July 2018.

[78] Lionel Pibre, Jérôme Pasquet, Dino Ienco, and Marc Chaumont. Deep Learning is a Good Steganalysis Tool when Embedding Key is Reused for Different Images, Even if There is a Cover Source-Mismatch. In *Proceedings of Media Watermarking, Security, and Forensics, MWSF'2016, Part of I&ST International Symposium on Electronic Imaging, EI'2016*, pages 1–11, San Francisco, California, USA, February 2016.

[79] Y. Qian, J. Dong, W. Wang, and T. Tan. Learning and Transferring Representations for Image Steganalysis Using Convolutional Neural Network. In *Proceedings of IEEE International Conference on Image Processing, ICIP'2016*, pages 2752–2756, Phoenix, Arizona, September 2016.

[80] Yinlong Qian, Jing Dong, Wei Wang, and Tieniu Tan. Deep Learning for Steganalysis via Convolutional Neural Networks. In *Proceedings of Media Watermarking, Security, and Forensics 2015, MWSF'2015, Part of IS&T/SPIE Annual Symposium on Electronic Imaging, SPIE'2015*, volume 9409, pages 94090J–94090J–10, San Francisco, California, USA, February 2015.

[81] Weize Quan, Kai Wang, Dong-Ming Yan, and Xiaopeng Zhang. Distinguishing Between Natural and Computer-Generated Images Using Convolutional Neural Networks. *IEEE Transactions on Information Forensics and Security*, 13(11):2772–2787, November 2018.

[82] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *Proceedings of the International Conference on Learning Representations, ICLR'2016*, page 16, Caribe Hilton, San Juan, Puerto Rico, May 2016.

[83] Pascal Schöttle and Rainer Böhme. A Game-theoretic Approach to Content-adaptive Steganography. In *Proceedings of the 14th International Conference on Information Hiding, IH'12*, volume 7692, pages 125–141, Berkeley, CA, USA, 2012.

[84] Pascal Schöttle and Rainer Böhme. Game Theory and Adaptive Steganography. *IEEE Transactions on Information Forensics and Security*, 11(4):760–773, April 2016.

[85] Vahid Sedighi and Jessica Fridrich. Effect of Imprecise Knowledge of the Selection Channel on Steganalysis. In *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2015, pages 33–42, Portland, Oregon, USA, 2015.

[86] Vahid Sedighi and Jessica J. Fridrich. Histogram Layer, Moving Convolutional Neural Networks Towards Feature-Based Steganalysis. In *Proceedings of Media Watermarking, Security, and Forensics, MWSF'2018, Part of IS&T International Symposium on Electronic Imaging, EI'2017*, pages 50–55, San Francisco, California, USA, February 2017.

[87] Vahid Sedighi, Rémi Cogranne, and Jessica Fridrich. Content-Adaptive Steganography by Minimizing Statistical Detectability. *IEEE Transactions on Information Forensics and Security, TIFS'2016*, 11(2):221 – 234, February 2016.

[88] Vahid Sedighi, Jessica J. Fridrich, and Rémi Cogranne. Toss that BOSSbase, Alice! In *Proceedings of Media Watermarking, Security, and Forensics, MWSF'2018, Part of IS&T International Symposium on Electronic Imaging, EI'2016*, pages 1–9, San Francisco, California, USA, February 2016.

[89] Haichao Shi, Jing Dong, Wei Wang, Yinlong Qian, and Xiaoyu Zhang. SSGAN: Secure Steganography Based on Generative Adversarial Networks. In *Proceedings of the 18th Pacific-Rim Conference on Multimedia, PCM'2017*, volume 10735 of *Lecture Notes*

in Computer Science, Springer, pages 534–544, Harbin, China, September 2017.

[90] Gustavus J. Simmons. The Subliminal Channel and Digital Signatures. In *Proceeding of Crypto'83*, pages 51–67, Santa Barbara, CA, August 1983. New York, Plenum Press.

[91] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceeding of International Conference on Learning Representations, ICLR'2015*, page 12, San Diego, CA, May 2015.

[92] Xiaofeng Song, Fenlin Liu, Chunfang Yang, Xiangyang Luo, and Yi Zhang. Steganalysis of Adaptive JPEG Steganography Using 2D Gabor Filters. In *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec'2015*, pages 15–23, Portland, Oregon, USA, June 2015.

[93] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR'2015*, pages 1–9, Boston, MA, USA, June 2015.

[94] Shunquan Tan and Bin Li. Stacked Convolutional Auto-Encoders for Steganalysis of Digital Images. In *Proceedings of Signal and Information Processing Association Annual Summit and Conference, APSIPA'2014*, pages 1–4, Chiang Mai, Thailand, December 2014.

[95] Weixuan Tang, Shunquan Tan, Bin Li, and Jiwu Huang. Automatic Steganographic Distortion Learning Using a Generative Adversarial Network. *IEEE Signal Processing Letters*, 24(10):1547–1551, October 2017.

[96] Weixuan Tang, Bin Li, Shunquan Tan, Mauro Barni, and Jiwu Huang. CNN-based Adversarial Embedding for Image Steganography. *IEEE Transactions on Information Forensics and Security*, 14(8), August 2019.

[97] Clement Fuji Tsang and Jessica J. Fridrich. Steganalyzing Images of Arbitrary Size with CNNs. In *Proceedings of Media Watermarking, Security, and Forensics, MWSF'2018, Part of IS&T International Symposium on Electronic Imaging, EI'2018*, pages 121(1)–121(8), Burlingame, California, USA, 28 January - 2 February 2018.

[98] Denis Volkhonskiy, Ivan Nazarov, Boris Borisenko, and Evgeny Burnaev. Steganographic Generative Adversarial Networks. Never Published, 2017.

[99] Chao Xia, Qingxiao Guan, Xianfeng Zhao, Zhoujun Xu, and Yi Ma. Improving GFR Steganalysis Features by Using Gabor Symmetry and Weighted Histograms. In *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'17, page 11, Drexel University in Philadelphia, PA, June 2017.

[100] Chao Xia, Qingxiao Guan, Xianfeng Zhao, Zhoujun Xu, and Yi Ma. Improving GFR Steganalysis Features by Using Gabor Symmetry and Weighted Histograms. In *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2017, pages 55–66, Philadelphia, Pennsylvania, USA, 2017.

[101] G. Xu, H. Z. Wu, and Y. Q. Shi. Structural Design of Convolutional Neural Networks for Steganalysis. *IEEE Signal Processing Letters*, 23(5):708–712, May 2016.

[102] Guanshuo Xu. Deep Convolutional Neural Network to Detect J-UNIWARD. In *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2017, pages 67–73, Drexel University in Philadelphia, PA, June 2017.

[103] Guanshuo Xu, Han-Zhou Wu, and Yun Q. Shi. Ensemble of CNNs for Steganalysis: An Empirical Study. In *Proceedings of the 4th ACM Workshop on Information Hiding*

and Multimedia Security, IH&MMSec'2016, pages 103–107, Vigo, Galicia, Spain, June 2016.

[104] Jianhua Yang, Danyang Ruan, Jiwu Huang, Xiangui Kang, and Yun-Qing Shi. An Embedding Cost Learning Framework Using GAN; (*previously named "spatial image steganography based on generative adversarial network" on ArXiv*). *IEEE Transactions on Information Forensics and Security, TIFS*, 15:839 – 851, June 2019.

[105] Jianhua Yang, Danyang Ruan, Xiangui Kang, and Yun-Qing Shi. Towards Automatic Embedding Cost Learning for JPEG Steganography. In *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2019, pages 37–46, Paris, France, July 2019.

[106] Jian Ye, Jiangqun Ni, and Y. Yi. Deep Learning Hierarchical Representations for Image Steganalysis. *IEEE Transactions on Information Forensics and Security, TIFS*, 12(11):2545–2557, November 2017.

[107] Mehdi Yedroudj, Marc Chaumont, and Frédéric Comby. How to Augment a Small Learning Set for Improving the Performances of a CNN-Based Steganalyzer? In *Proceedings of Media Watermarking, Security, and Forensics, MWSF'2018, Part of IS&T International Symposium on Electronic Imaging, EI'2018*, page 7, Burlingame, California, USA, 28 January - 2 February 2018.

[108] Mehdi Yedroudj, Frédéric Comby, and Marc Chaumont. Yedrouj-Net: An Efficient CNN for Spatial Steganalysis. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'2018*, pages 2092–2096, Calgary, Alberta, Canada, April 2018.

[109] Mehdi Yedroudj, Frédéric Comby, and Marc Chaumont. Steganography using a 3 player game. In *under submission*, volume abs/1907.06956, 2019. URL `http://arxiv.org/abs/1907.06956`.

[110] Yassine Yousfi, Jan Butora, Jessica Fridrich, and Quentin Giboulot. Breaking ALASKA: Color Separation for Steganalysis in JPEG Domain. In *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2019, pages 138–149, Paris, France, July 2019.

[111] Ahmad Zakaria, Marc Chaumont, and Gérard Subsol. Quantitative and Binary Steganalysis in JPEG: A Comparative Study. In *Proceedings of the European Signal Processing Conference, EUSIPCO'2018*, pages 1422–1426, Roma, Italy, September 2018.

[112] Ahmad Zakaria, Marc Chaumont, and Gérard Subsol. Pooled Steganalysis in JPEG:how to deal with the spreading strategy? In *Proceedings of the IEEE International Workshop on Information Forensics and Security, WIFS'2019*, page 6, Delft, The Netherlands, December 2019.

[113] J. Zeng, S. Tan, B. Li, and J. Huang. Large-Scale JPEG Image Steganalysis Using Hybrid Deep-Learning Framework. *IEEE Transactions on Information Forensics and Security*, 13(5):1200–1214, May 2018.

[114] Jishen Zeng, Shunquan Tan, Bin Li, and Jiwu Huang. Pre-Training via Fitting Deep Neural Network to Rich-Model Features Extraction Procedure and its Effect on Deep Learning for Steganalysis. In *Proceedings of Media Watermarking, Security, and Forensics 2017, MWSF'2017, Part of IS&T Symposium on Electronic Imaging, EI'2017*, page 6, Burlingame, California, USA, January 2017.

[115] R. Zhang, F. Zhu, J. Liu, and G. Liu. Depth-Wise Separable Convolutions and Multi-Level Pooling for an Efficient Spatial CNN-Based Steganalysis; (*previously named*

"efficient feature learning and multi-size image steganalysis based on cnn" on ArXiv). *IEEE Transactions on Information Forensics and Security, TIFS*, 2020.

[116] Yiwei Zhang, Weiming Zhang, Kejiang Chen, Jiayang Liu, Yujia Liu, and Nenghai Yu. Adversarial Examples Against Deep Neural Network Based Steganalysis. In *Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec'2018, pages 67–72, Innsbruck, Austria, June 2018.

[117] Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. HiDDeN: Hiding Data With Deep Networks. In *Proceedings of the 15th European Conference on Computer Vision, ECCV'2018*, volume 11219 of *Lecture Notes in Computer Science, Springer*, pages 682–697, Munich, Germany, September 2018.

ACKNOWLEDGMENTS

I would like to thank the PhD students (and the Masters' students) who directly or indirectly worked on the topic: Sarra Kouider, Amel Tuama, Jerôme Pasquet, Hasan Abdulrahman, Lionel Pibre, Mehdi Yedroudj, Ahmad Zakaria, during the period (2015-2018). Without all of them, this chapter would never have been possible.

I would also like to thank my two colleagues, Frédéric Comby and Gérard Subsol, who helped me supervise this nice small-world.

I thank the French working group, Caroline Fontaine, Patrick Bas, Rémi Cogranne, with whom I have many interesting discussion and who encourage me to write this chapter. I would like to thank the Direction Générale de l'Armement (DGA) for its support on steganalysis through the Alaska project ANR (ANR-18-ASTR-0009).

I thank the LIRMM (the lab), ICAR (my team - with all the members), the Montpellier University and the Nîmes university, HPC@LR, for all the given resources which allowed me to run such a work.

Finally, I would like to thank my wife, Nathalie, my four little smurfs, Noam, Naty, Coline, Mila, and Louis, who are the guardian of my sanity : −)

AUTHOR BIOGRAPHY

Marc CHAUMONT is Associate Professor (HDR Hors-Classe) accredited to supervise research, at the LIRMM laboratory (Laboratory of Computer Science, Robotics and Microelectronic), University of Montpellier and University of Nîmes, in France. He is a member of the IEEE Signal Processing - Information Forensics Security - Technical Committee and is a reviewer for the major conferences and journals related to steganography/steganalysis. He joined the LIRMM in September 2005. He received his PhD in Computer Sciences at IRISA Rennes in 2003. His main research interests are in steganography, steganalysis, digital image forensic, and objects detection with Deep Learning.