Deep Learning in steganography and steganalysis from 2015 to 2018
Marc Chaumont

To cite this version:

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.
Deep Learning in steganography and steganalysis from 2015 to 2018

Marc CHAUMONT,a,*

*Montpellier University, LIRMM (UMR5506) / CNRS, Nîmes University, France,
LIRMM/ICAR, 161, rue Ada, 34392 Montpellier, France
aCorresponding: 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, 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, 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.

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) followed by a classification using a classifier (EC). 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). During the period 2015 - 2018, many publications have shown that it is possible to 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.

### 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).
1.1 The building blocks of a deep neuronal network

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

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.

Figure 1.1 Yedroudj-Net network [108].

GitHub link on Yedroudj-Net: https://github.com/yedmed/steganalysis_with_CNN_.
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

The pre-processing module is used to filter the input images. A high-pass filter is applied to the image to enhance the high-frequency components. The filter is given by the equation:

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

(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.

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.
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)\cite{34}, a statistical moments extractor\cite{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\cite{108} network, the Zhu-Net\cite{115} network, or the Tsang et al. network\cite{97}.

Also note that\cite{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\cite{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 Section 1.2.3), 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
The different steps of the convolution module

1.2 The different steps of the convolution module

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 $F^{(l)}_k$ be the $k^{th}$ ($k \in \{1, ..., K^{(l)}\}$) 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 $l^{th}$ block. The convolution within the pre-processing block with the $k^{th}$ filter results in a filtered image, denoted $\tilde{I}^{(1)}_k$, such that:

$$\tilde{I}^{(1)}_k = I^{(0)} \ast F^{(1)}_k. \quad (1.2)$$

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 $I^{(l-1)}_k$ with $k = \{1, ..., K^{(l-1)}\}$.

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

$$\tilde{I}^{(l)}_k = \sum_{i=1}^{i=K^{(l-1)}} I^{(l-1)}_i \ast F^{(l)}_{k,i}, \quad (1.3)$$

with $\{F^{(l)}_{k,i}\}_{i=1}^{i=K^{(l-1)}}$ 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.

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 $1 \times 1$ 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-
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:

\[
I_k^{(l)} = \text{norm} \left( \text{pool} \left( f \left( b_k^{(l)} + \sum_{i=1}^{K^{(l-1)}} I_i^{(l-1)} \ast F_{k,i}^{(l)} \right) \right) \right),
\]  

(1.4)

where \(b_k^{(l)} \in \mathbb{R}\) the scalar standing for the convolution bias, \(f()\) the activation function applied pixel by pixel on the filtered image, \(\text{pool}()\), 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 \(f(x) = \sin(x)\), a Gaussian function as in \[80\], \(f(x) = e^{-x^2/\sigma^2}\), 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
1.2 The different steps of the convolution module

Activation Function

![Activation Function Diagram]

Derivative activation functions

![Derivative Activation Functions Diagram]

Figure 1.3 Three main activation functions and their derivatives.

also avoided, since this type of function can cause the value of the backpropagated 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.
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:

$$BN(x, \gamma, \beta) = \beta + \gamma \frac{x - E[X]}{\sqrt{Var[X]} + \epsilon},$$  \hspace{1cm} (1.5)
with $E[X]$ the expectation, $Var[X]$ the variance, and $\gamma$ and $\beta$ 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 $\gamma$ and $\beta$ 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 $\gamma$ and $\beta$ 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 × 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] (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.

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 backpropagation 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
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 example 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 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?

---

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 [91] 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] 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 [114], 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.

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
1.5 The different networks used over the period 2015-2018

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 [107, 108] (See Figure 1.10), ReST-Net [62] (See Figures 1.7 and 1.8), SRNet [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 $256 \times 256$ 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](https://github.com/yedmed/steganalysis_with_CNN_Yedroudj-Net)
8 SRNet source code: [https://github.com/Steganalysis-CNN/residual-steganalysis](https://github.com/Steganalysis-CNN/residual-steganalysis)
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
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 $256 \times 256$ 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 v1.01 512 × 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
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 \( 3 \times 25 \) 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
1.5 The different networks used over the period 2015-2018

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 [23], a “point” high-pass filter (referenced as “catalyst kernel”) which complements the “SQUARE5×5”, and two directional Gabor Filters (angles 0 and π/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 DCT’s 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 $\delta_{uSA}^{1/2}$ \[6\] 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/](https://alaska.utt.fr/)
1.5 The different networks used over the period 2015-2018

The use of a single CNN with millions of images 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, 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 (a class is associated with each embedding algorithm - there is thus a latent partition). Note that this idea, among others, has been used by the winners of the Alaska challenge. 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 steganalyzer 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, IMFA, CFT, where the idea is to define an invariant latent space. Another approach is ATS 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. 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 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. 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].](image)

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

---

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:

---

• 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
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 $64 \times 64$ pixels. 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 $P_e = 31.3\%$ (note that a $P_e$ of 50% is equivalent to a random detector) on a $256 \times 256$ pixels BOSS database, for a payload size of 0.1 bpp. Due to the square root law, the $P_e$ would be higher for the $64 \times 64$ 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 $100 \times 3$ 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. 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, and modification approaches such as S-UNIWARD, HILL, MiPod or even Synch-Hill, but this margin has been reduced. 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 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, and UT-6HPF-GAN; 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 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, 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 enriched with 6 high-pass filters for the pre-processing in the same spirit as Ye-Net or Yedroudj-Net.

The system learns on a first database, and then security comparisons are made on the 256 × 256 pixels BOSS, LIRMMBase, and BOWS2.

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.
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 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 [70]). During the iterations, the cost map is updated, but there is only a $\beta$ 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 $\beta$ 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 $\beta$ 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 $\beta$ 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 \textit{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 \textit{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 \textit{strategic equilibrium} since the problem is expressed as a min-
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

---

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\textsuperscript{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.

\section*{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?

\section*{REFERENCE}

\begin{flushleft}
and thus a rate of 0.3 bpp for these images.
\end{flushleft}


1.6 Steganography by deep-learning


1.6 Steganography by deep-learning


[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,
1.6 Steganography by deep-learning

July 2018.


CHAPTER 1 Deep Learning in steganography and steganalysis


1.6 Steganography by deep-learning


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.