# Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover source-mismatch

Lionel Pibre, Jérôme Pasquet, Dino Ienco, Marc Chaumont

# Deep Learning for steganalysis is better than a Rich Model with an Ensemble Classifier, and is natively robust to the cover source-mismatch

**Lionel PIBRE** [2,3], **Jérôme PASQUET** [2,3], **Dino IENCO** [2,3], **Marc CHAUMONT** [1,2,3]

[1] **UNIVERSITE DE NIMES, 30 021 Nîmes Cedex 1, France**
[2] **UNIVERSITE MONTPELLIER, UMR5506-LIRMM, 34 095 Montpellier Cedex 5, France**
[3] **CNRS, UMR5506-LIRMM, 34 392 Montpellier Cedex 5, France**
{lionel.pibre, jerome.pasquet, dino.ienco, marc.chaumont}@lirmm.fr

## Abstract

*Since the BOSS competition, in 2010, most steganalysis approaches use a learning methodology involving two steps: feature extraction, such as the Rich Models (RM), for the image representation, and use of the Ensemble Classifier (EC) for the learning step. In 2015, Qian et al. have shown that the use of a deep learning approach that jointly learns and computes the features, is very promising for the steganalysis.*

*In this paper, we follow-up the study of Qian et al., and show that, due to intrinsic joint minimization, the results obtained from a Convolutional Neural Network (CNN) or a Fully Connected Neural Network (FNN), if well parameterized, surpass the conventional use of a RM with an EC.*

*First, numerous experiments were conducted in order to find the best "shape" of the CNN. Second, experiments were carried out in the clairvoyant scenario in order to compare the CNN and FNN to an RM with an EC. The results show more than 16% reduction in the classification error with our CNN or FNN. Third, experiments were also performed in a cover-source mismatch setting. The results show that the CNN and FNN are naturally robust to the mismatch problem.*

*In Addition to the experiments, we provide discussions on the internal mechanisms of a CNN, and weave links with some previously stated ideas, in order to understand the impressive results we obtained.*

## 1.Introduction

The state-of-the-art for steganalysis currently consists of using a two-step machine-learning methodology.

The first step requires the extraction of features describing the image. Those features must be *diverse* [1], which means that they should capture the maximum of information modeling the image, and they also should be *complete* [2], which means that their values should be different between a cover and a stego. The best feature set to represent an image has so far been supplied by Rich Models [3].

The second step consists of learning to distinguish cover model from the stego(s) model(s). Depending on the memory or computation requirements, the steganalyst can use an SVM [4], an Ensemble Classifier [5], or a Perceptron [6].

When analyzing the empirical security of an embedding al-

gorithm in a laboratory environment [7], we select the "clairvoyant scenario" [8], i.e. we suppose that the steganalyst knows the algorithm and the payload size that has been used by the steganographer, and has good knowledge of the cover distribution through a set of images of the same type as those used by the steganographer.

Until 2015, in the clairvoyant scenario, the best classifier was the Ensemble Classifier [5]. This classifier is able to treat high dimensional vectors, is easily parallelizable, and has a smaller computational complexity than SVM. Moreover, some improvements have been proposed in order to increase its efficiency, such as the use of embedding probabilities in order to better steganalyze the adaptive algorithms [9], tuning of false alarm probability [10], or treating the cover-source mismatch problem [11], where the best classifier is also the Ensemble Classifier [12].

Yet, in recent years, in different areas, the use of deep learning networks challenges traditional two step approaches (feature extraction, and use of a classifier) [13].

In the steganalysis field, Qian *et al.* [14] proposed, in 2015, to use deep learning to replace the traditional two step approach. In their article, Qian *et al.* obtained a detection percentage of only 3% to 4% lower than that obtained with the Ensemble Classifier [5], and SRM features [3]. The tested algorithms were HUGO [15], WOW [16], and S-UNIWARD [17], on the BOSS-base database [18]. Those first results were encouraging since the study was only a proof of concept, and because the feature vector dimension did not compare favorably with respect to deep learning. Indeed, the dimension of the feature vector of the last convolution layer (layer 5) provides only 256 features, whereas the SRM (Spatial Rich Models) dimension of the feature vector provides 34671 [3].

In this article, we pursue the study of the steganalysis via deep learning. After many months of experiments, we obtained a reduction of more than 16% in the classification error compared to the state of the art. We also found a network that is robust to the cover-source mismatch.

The network we built is very different from that of Qian *et al.* [14]. In Section 2, we review the major concepts of a Convolutional Neural Network. In Section 3, we introduce the experimental settings, describe the "shape" of our best CNNN, and we present steganalysis results in both scenarios: clairvoyant and cover-source mismatch. Finally, in Section 4, we discuss the link

between the network construction and steganalysis research, and we explain our remarkable results.

## 2. Convolutional Neural Network

Neural networks have been studied since the fifties. Initially, they were proposed to model the brain behavior. In computer science, especially in artificial intelligence, they have been used for 30 years for learning purposes. Until recently [19], neural networks were considered as having a too long learning time, and as being less efficient than modern classifiers.

Recently, due to recent advances in the neural network field [20], and to the computational power supplied by GPUs, deep learning approaches have been proposed as a natural extension of neural networks, and they are getting popular due to their high classification performance. Deep learning networks are big neural networks that can directly take data as input. In image processing, the network is directly fed with pixels. A deep learning network handles two steps at once (feature extraction and classification). Since 2006 [19], many adjustments have been proposed to improve the robustness and reduce the computational costs.

In this paper, we recall the major concepts of a Convolutional Neural Networks (CNN), which is a deep learning network that has proved its efficiency in image classification competitions [13], and that was used by Qian *et al.* for steganalysis purposes.

The learning methodology is similar to the classical one. An image database is needed, with, for each image, its label (i.e. its class). Each image is given as network input; in that case, each pixel value is taken as input of one or many neurons. The network is made of a given number of *layers*. A layer consists of neurons that take input values, do some computations, and then returns values that are supplied to the next layer.

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

As an illustration, Figure 1 gives the network used by Qian *et al.* For this network, an image of size $256 \times 256$ is first filtered with a high-pass filter whose kernel is denoted $F^{(0)}$, and size is $5 \times 5$ (see Eq. 1). Note that this preliminary mandatory step is specific to the steganalysis problem. We observed that CNNs do not converge without this preliminary high-pass filtering. Then, the filtered image, of size $252 \times 252^1$, is given to the first layer. In the Qian *et al.* network (see Figure 1), there are 5 convolution layers.

Let us now explain more precisely what a layer is in a Convolutional Neuronal Network.

### 2.1. Layer

A layer is made of neurons that take input values, do some computations, and then returns values that are supplied to the next layer. More precisely, inside a layer, computations are done in

three successive steps: a *convolution* step (see Section 2.2), the application of an *activation function* (see Section 2.3), and then a *pooling* step (see Section 2.4). Note that the outputs of a layer could be considered as a *set* of images. In CNN terminology, each image is named a *feature map*.

In Fig. 1, representing the Qian *et al.* network, the first layer generates 16 *feature maps*, each of size $124 \times 124$. Note that this means that there are 16 filters and thus 16 convolutions which are applied to the input image of size $252 \times 252$. From the second to the fifth layer, there are the same three steps: convolution, activation, and pooling, but this time the convolutions are applied to all feature maps. We discuss in further detail how the convolutions and sub-sampling are fulfilled in the next subsection.

The last convolution layer is connected to a fully connected two layer neuronal network. Then, a softmax function is connected to the outputs of the last layer in order to normalize the two outputs delivered by the network between $[0, 1]$. The softmax function gives the predicted probability of belonging to a class, while knowing the weighted outputs from the last layer. Thus, 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 obtained by returning the class with the highest probability.

A Convolutional Neural Network, similar to a classical neuron network, needs a long learning time in order to tune each unknown parameter. In the case of Qian *et al.* network, the number of unknown parameters is close to 63 000, which indeed is not very large since convergence requires less than 2 hours with GPU programming on a Nvidia Tesla K80. Learning is achieved with the well known *back-propagation algorithm*. Roughly speaking, back-propagation of the error is equivalent to a gradient descent which is a well-known function optimization technique.

Network learning can thus be seen as the optimization of a function, with lots of unknown parameters, through the use of a well thought stochastic gradient descent. Due to the huge number of parameters to learn, the neural network needs a database that has a considerable number of examples in order to converge. Moreover, the database examples must be diverse enough to obtain a good generalization of the network.

Let us now explain each step inside a layer of a Convolutional Neural Network more in detail: convolution, activation, and pooling.

### 2.2. Convolution

For a given layer and a set of *feature map* as input, the first processing consist of applying the convolutions.

For the first layer, the convolution is trivial since there is only one image as input. Convolution is done between the input image and a filter. In the Qian *et al.* network, there are 16 filters (see Figure 1). Each filter leads to a filtered image. Then the second (activation function; see Section 2.3), and third (pooling; see Section 2.4) steps are applied, leading to a new image named a *feature map*. In the Qian *et al.* network, there are 16 *feature maps* at the output of the first layer.

Formally, let $I^{(0)}$ denote the image given to the CNN (note that for the Qian *et al.* network, image $I^{(0)}$ is a high-pass filtered image; see Section 2 and Figure 1). Let $F_k^{(l)}$ denote the $k^{th}$ filter
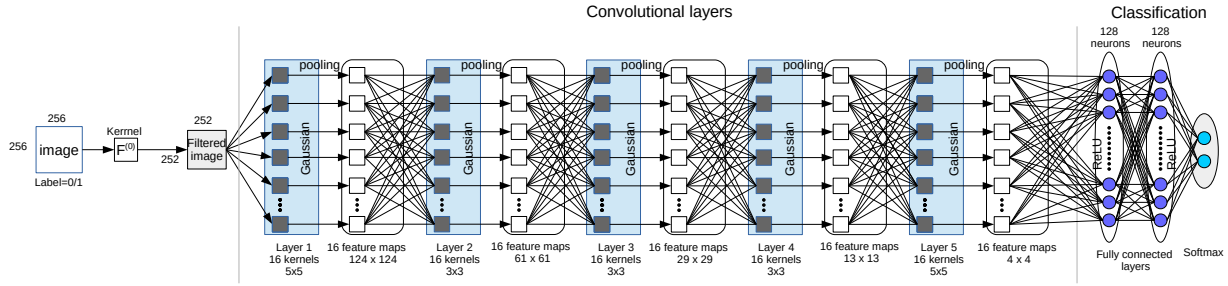
---

[1] The filtered image is smaller than the original image because there is no padding.

**Figure 1.** *Qian et al. Convolutional Neural Network [14].*

from layer $l = \{1, ..., L\}$, with $L$ beeing the number of convolutional layers, and $k \in \{1, ..., K^{(l)}\}$, with $K^{(l)}$ beeing the number of filters of the $l^{th}$ layer ($K^{(l)}$ is also the number of feature map outputs by the $l^{th}$ layer). A convolution from the first layer with the $k^{th}$ filter leads to a filtered image, denoted $\tilde{I}_k^{(1)}$, such that:

$$\tilde{I}_k^{(1)} = I^{(0)} \star F_k^{(1)}. \tag{2}$$

Figure 2 gives an example of 64 filter kernels obtained with the most efficient network we obtained. Note that the filter looks like oriented band-pass filters.
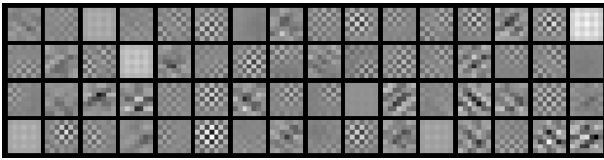


**Figure 2.** *64 filter kernels of size $7 \times 7$ from the first convolutional layer obtained for our most efficient CNN.*

From the second layer to the last convolution layer, the "convolution" is less classical since there are $K^{(l-1)}$ *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 $k^{th}$ filtered image, $\tilde{I}_k^{(l)}$, resulting from the convolution layer numbered $l$, is in fact the sum of $K^{(l-1)}$ convolutions, such that:

$$\tilde{I}_k^{(l)} = \sum_{i=1}^{i=K^{(l-1)}} I_i^{(l-1)} \star F_{k,i}^{(l)}, \tag{3}$$

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 for each convolution. There is no similar operation in the classical feature extraction process supplied by Rich Models [3], and to the best of our knowledge, in spatio-frequential decomposition. Figure 3 gives an example of the 1024 filter kernels obtained by the second layer of the most efficient network we obtained.

Remember that a convolution layer is made of three steps i.e. convolution, activation, and pooling. These three consecutive steps can be summarized by looking at the link between a feature
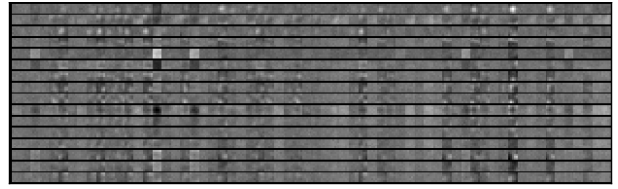


**Figure 3.** *The $16 \times 64$ filter kernels of size $5 \times 5$ from the second convolution layer obtained for our most efficient CNN.*

map from a layer to the previous one:

$$I_k^{(l)} = pool\left( f\left( b_k^{(l)} + \sum_{i=1}^{i=K^{(l-1)}} I_i^{(l-1)} \star F_{k,i}^{(l)} \right) \right), \tag{4}$$

with $b_k^{(l)} \in \mathbb{R}$ being a scalar used for setting a bias to the convolution, $f()$ being the activation function applied pixel per pixel to the filtered image, and the pooling, $pool()$, which pools a local neighborhood (see Section 2.4).

Note that the filter kernels (also referred as weights), and the bias have to be learned and are modified during back-propagation of the error. Thus, for a layer $l \in \{1, ..., L\}$, and for a number of filters per layer, $K^{(l)}$, with a convolution kernel made of $|F^{(l)}|$ weights, there is a total number of

$$\sum_{l=1}^{l=L} K^{(l)} \times (1 + |F^{(l)}|)$$

unknown parameters to be tuned during back-propagation for the convolutional layer part. As an example, for the Qian *et al.* network, the number of parameters coming from the five convolution layers is equal to 16 filters $\times (1 + 5 \times 5) + 3 \times (1 + 3 \times 3 \times 16) + (1 + 5 \times 5 \times 16)) = 13792$ unknown parameters coming from the convolution weights and the bias (and a total of 63456 unknown parameters for the entire network if we include the fully connected layers and the softmax).

Note that the complexity computation (i.e. the number of multiplications, or the number of addition minus 1) for the convolution layers of the network is close to[2]:

$$\sum_{l=1}^{l=L} K^{(l)} \times |I^{(l-1)}| \times (1 + |F^{(l)}|),$$

---

[2]The image borders are not always filtered, which actually gives a smaller number of operations.

with $K^{(l)}$ being the number of "convolutions" per layer, $|I^{(l-1)}|$, the size of a feature map from layer $l-1$, and $|F^{(l)}|$ the number of weights for a convolution. Note that this complexity does not take into account the *activation operation cost*, the *pooling operation cost*, and the *normalization operation cost*.

Those computations are very classical and can be accelerated, for example through a fast convolution in the Fourier domain, or through the use of parallel computation. Thus, when using the network, if the size of the input image, the number of filters, the size of the filters, and the number of layers are not too big, the convolution computation cost, with the pooling and the normalization included, is more or less $O(L \times K^{(0)} \times I^{(0)} \times |F^{(2)}|)$, which is not very high.

As an example, for the Qian *et al.* network, this complexity for the convolution part of the network would give more or less $5 \times 16 \times (252 \times 252) \times (3 \times 3 \times 16)$ additions or multiplications, which is less than 700 Mega operations. This is very small, for example, for a CPU Intel Core i7 which can compute at more than 50 GigaFLOPS, or for a GPU Nvidia Tesla K80 which can compute at more than one TeraFLOPS. Even when looking at the entire network, the computational cost is not very high. The long learning times are due to the fact that those operations have to be done on a big database, which has to be scanned many times in order that the back-propagation process does converge the network.

### 2.3.Activation

Once each convolution from a convolution layer has been applied, an *activation* function, $f()$ (see Eq. 4), is applied to each value of the filtered image, $\tilde{I}_k^{(l)}$ (Eq. 2 and Eq. 3). This function is named the activation function in reference to the notion of binary activation in the first network neuron definition. The activation function may for example be an absolute function $f(x) = |x|$, a sine function $f(x) = sinus(x)$, a Gaussian function as in the Qian *et al.* network $f(x) = \frac{e^{-x^2}}{\sigma^2}$, a ReLU (for Rectified Linear Units): $f(x) = max(0, x)$, etc...

Those functions break the linearity property resulting from the linear filtering done during the convolutions. This is usually an interesting property that is exploited in the Ensemble Classifier through the majority vote [5], and that is also used in the Rich Models with the Min-Max features [3]. The choice of the activation function is linked to the classification problem. For example, Qian *et al.* proposed to use an unusual Gaussian function. Note that the chosen function should be derivable in order to compute the back-propagation error. The derivative could be more or less computationally costly, which has an impact on the learning time. The choice of the activation function is thus often guided by this computational criteria. During the experiments, we observed that the best results were obtained with the ReLU activation function.

### 2.4.Pooling

The pooling operation consists of computing the *average* or *maximum* on a local neighborhood. In the object classification field, this operation, and especially the use of a maximum, ensures a translation invariance of the features. It was introduced in order to reduce the variance obtained during convolution. Qian *et al.* propose to use the average operation because the stego noise is

very small. We also empirically validated this fact in our experiments. The results obtained using the average outperform the one obtained by the maximum operation.

Moreover, the pooling is coupled with a sub-sampling operation in order to reduce the size of the obtained feature map in comparison to the size of those of the previous layer. In the article of Qian *et al.*, there is a reduction factor of four between the feature map size of a layer and that from the previous layer. This can then be seen as a classical down-sampling with preliminary low pass filtering. This is useful to reduce the memory occupation in the GPU. Nevertheless, this step is similar to denoising, and from a signal processing point of view, it induces information loss. This pooling step does not seem interesting for a steganalysis, and indeed we experimentally observed that suppressing the pooling operation increased the classification results by more than 8%. In return, suppressing the pooling step gives feature maps of a constant size in all the layers, and this leads to an increase in the computational cost, and an increase in GPU memory consumption.

## 3.Experiments

### 3.1.The databases and learning settings used

For the experiments, we first took the database BOSSBase v1.0 [18] consisting of 10 000 grey-level images of size $512 \times 512$ coming from 7 different cameras, then we split each image in four in order to obtain 40 000 images of size $256 \times 256$. We named this database the *cropped BOSSBase database*. In their article, Qian *et al.* also reduced the images sizes due to the GPU memory limitation. Note that they applied image resizing instead of the image cropping.

We also created a second database that we named LIRMM-Base[3] ⚘LIRMMBase. The LIRMMBase database consists of 1008 grey-level images, coded on 8 bits, of size $256 \times 256$. There are 6 cameras, none of them present in BOSSBase, and 168 images per camera. This database allows evaluation of the cover-source mismatch phenomenon. Note that readers may find different versions of the LIRMMBase on its hosting website: the color version, LIRMMBaseColor, with 15320 images, the $512 \times 512$ grey-level image database, LIRMMBase512x512, with 252 images, and the LIRMMBase256x256 database with 1008 grey level images. The images come from well-known databases i.e. Columbia, Dresden, Photex, and Raise databases. Note that we used the same script as that used for the BOSSBase in order to transform the RAW full resolution color images into grey-level images[4].

---

[3] ⚘LIRMMBase If researchers use this free-access database, they are required to cite as follows: " LIRMMBase: A database built from a mix of Columbia, Dresden, Photex, and Raise databases, and whose images do not come from the same cameras as the BOSSBase database ; For the first time used in [ref]. " L. Pibre, J. Pasquet, D. Ienco, and M. Chaumont, LIRMM Laboratory, Montpellier, France, June 2015, Website: `www.lirmm.fr/~chaumont/LIRMMBase.html`. [ref] Lionel Pibre, Jérôme Pasquet, Dino Ienco, and Marc Chaumont, "Deep Learning for steganalysis is better than a Rich Model with an Ensemble Classifier, and is natively robust to the cover source-mismatch," *in Proceedings of Media Watermarking, Security, and Forensics, Part of IS&T International Symposium on Electronic Imaging, EI'2016*, San Francisco, California, USA, 14-18 Feb. 2016, 10 pages.

[4]The script converts the full resolution RAW color images into RGB images, then resizes the images such that the smaller side is 512 pix-
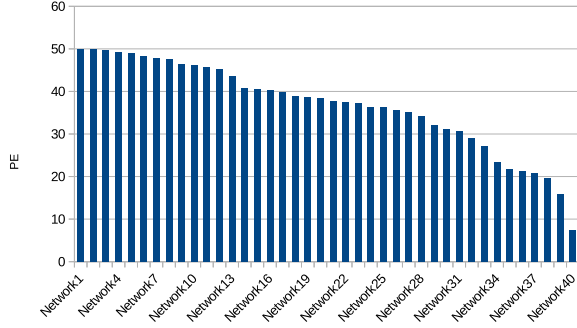
**Figure 4.** *Probabilities of error for some of the tested CNN.*



**Figure 5.** *A Convolutional Neural Network (CNN) greater in height than in depth, and without pooling.*

In our experiments, we embedded the messages (using the simulator) with S-UNIWARD [17] at 0.4 bits per pixel. After embedding, the database obtained from BOSSBase consisted of 80 000 images (40 000 covers and 40 000 stegos), and the database from LIRMMBase consisted of 2 016 images (1008 covers and 1008 stegos). We limited our experiments to this payload size due to the high number of computations, and experiments that we have led on the CNN. More than 40 CNN were tested with many parameter variations. Figure 4 illustrates the different values of the probability of error for some of the tested CNN.

Due to the high number of parameters, it is necessary to have a high number of iterations of the back-propagation process in order for the CNN to converge. In our experiments, the number of times the database is scanned was between 100 and 200. With a learning database consisting of 60 000 images of size $256 \times 256$ this leads to a learning time that is less than one day (sometimes 2 hours, sometimes more, depending on the parameters and the charge on the GPU) with the most efficient double precision GPU card on the market on June 2015, i.e. the Nvidia Tesla K80. It thus takes more than one-month of computation in order to find our "best" network. For a reproducibility of experiments, the main parameters are given: the "mini-batch" size is 128, the "moment" is 0.9, the "learning coefficient" is 0.001 for weights and 0.002 for bias, the "weight decay" is 0.004 for convolutions layers and 0.01 for the fully connected network, the "drop out" is not activated.

The paper essentially demonstrates that a CNN is more efficient than an Ensemble Classifier, or an Ensemble Classifier informed on the selection channel (adaptive steganalysis scenario) [21, 9]. Moreover, the paper also shows that CNN exhibits a surprising invariance property with respect to the cover-source mismatch, which is currently, a phenomenon that we are not fully able to explain. For our defense, the deep learning performances in many other topics are not well explained.

### 3.2. The shape of our best CNN

We tested many CNN (as shown in Figure 4), and the most efficient network we obtained consisted of only two convolutional layers, followed by a three layer fully connected network. Figure 5 illustrates this network. The input image, of size $256 \times 256$, is first high-pass filtered with the same filter as Qian *et al.* (see Eq.

els long, then crops the images to $512 \times 512$, and finally converts them to greyscale 8-bit images; the script may be found at `www.lirmm.fr/~chaumont/LIRMMBase.html`.
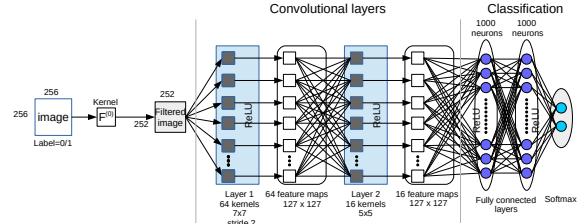
1; and discussion in Section 2). The size of the filtered image is thus $252 \times 252$.

For the first layer, 64 filters of size $7 \times 7$ are applied. Figure 2 illustrates those 64 filters after a learning on 60 000 images (30000 covers and their 30000 associated stegos) from the cropped BOSSBase. As already stated, those filters seem to act as oriented band-pass filters. Note that, for memory constraints, convolution is only applied 1 pixel above 2 for lines and columns (the "stride" parameter is equal to 2) which leads to filtered images of size $127 \times 127$[5]. Note that in comparison to the Qian *et al.* network, we drastically increased the number of filters in the first layer and reduced the number of CNN layers. The increase in height is one of the reasons why our network generates better results compared to the Qian *et al.* architecture. The increase in layer number leads to a loss of information, probably due to the negative impact of the pooling step (sub-sampling).

After 64 convolutions, the ReLU activation function is applied (see Section 2.3); this function forces the values to be positive. Note that the use of a Gaussian activation function, as in the article of Qian *et al.*, does not improve the results. As already discussed, the activation breaks the linearity of the successive convolutions applied during the traverse of the convolutional layers.

Note that we suppressed pooling because this step was counter-productive. The 64 feature maps returned by the first layer, are thus of size $127 \times 127$. Finally, the last process done by the layer is a normalization of the feature maps applied on each value, i.e on each position $(x, y)$ of $I_k^{(1)}$ for each $k \in \{1, ..., K^{(1)}\}$ with $K^{(1)} = 64$. The normalization is done across the maps, which is useful when using unbounded activation functions such as ReLU[6]:

$$norm(I_k^{(1)}(x,y)) =$$

$$\frac{I_k^{(1)}(x,y)}{\left(1 + \frac{\alpha}{size} \sum_{k'=max(0,k-\lfloor size/2 \rfloor)}^{k'=min(K,k-\lfloor size/2 \rfloor+size)} (I_{k'}^{(1)}(x,y))^2\right)^{\beta}}$$

---

[5]For this first layer, the convolution with a filter size $7 \times 7$ on an image size of $252 \times 252$ virtually leads to an image size of $253 \times 253$, because we selected the *padding* option and because the *stride* value of 2 implies image down-sampling, which gives an image size of $127 \times 127$.

[6]The normalization is done with a "local response normalization layer". *This enables detection of high-frequency features with a high neuron response, while damping responses that are uniformly large in a local neighborhood. This is a type of regularizer that encourages "competition" for big activities among nearby groups of neurons.* From the Cuda-convnet documentation.

with $\alpha$ and $\beta$, and *size*, the parameter set with default values[7].

The 64 feature maps of size $127 \times 127$ entering in the second layer are padded in order to obtain feature maps of size $131 \times 131$. Next, 16 "convolutions" are applied, as explained in Section 2.2. Equation 3 recalls this particular convolution step, where each sum of convolutions can be seen as the research of local signals in each of the feature maps through computation of the correlations between the researched signals and the feature maps. This step, and the normalization steps, probably explain the robustness to the cover-source mismatch that we observed during the experiments.

Figure 3 illustrates those $16 \times 64$ filters after a learning on 60 000 images (30000 covers and their 30000 associated stegos) from the cropped BOSSBase. ReLU activation is employed after applying the convolutions with those filters, and then the normalization is performed again. This leads to 16 feature maps of size $127 \times 127$ each. Note that concatenating all feature maps values leads to a feature vector consisting of 258 064 features, which is 7 times more than the SRM which consists of 34 671 features[8]

After the two convolution layers, there is a fully connected three layer neural network. The first and the second layers consist of 1000 neurons, and the last layer only has 2 neurons. The operations carried out in the first and second layers are dot products, bias additions, and the applications of the ReLU activation function. The operations performed in the last layer are dot products, bias additions, and then a softmax in order to rescale the output values in $[0,1]$.

### 3.3. Clairvoyant scenario

Our first test is in the clairvoyant scenario. In this scenario, we put forward the hypothesis that the steganalyst knows the embedding algorithm, has good knowledge of the statistical distribution of the image database used by the steganograph, and knows the relative payload size. This scenario almost matches the Kerckhoffs principle. The steganalyst knows all the public parameters (in the clairvoyant scenario, the selection channel is assumed to be inaccessible), and does not know the private parameters such as the embedding secret keys. This scenario is a laboratory scenario used to empirically assess the security of a steganographic embedding algorithm [7].

Our tests were carried out on the cropped BOSSBase database, which consists of 40 000 grey-level images on 8 bits and whose size is $256 \times 256$. We embed with S-UNIWARD [17] at 0.4 bits per pixels. The obtained set of images is thus made of 80 000 images (40 000 covers and 40 000 stegos). We limited our experiments to a single payload size due to the high number of computations, and the high number of experiments on the CNNs.

Three steganalysis approaches were evaluated. The first steganalysis was done using Rich Models [3] and an Ensemble Classifier [5]. The Rich Models is the SRM whose dimension is 34671. We will denote this steganalysis, **RM+EC**, for Rich Models and Ensemble Classifier. The second steganalysis was done using the most efficient CNN we built (see Fig. 5). We denote this steganalysis, **CNN**, for Convolutional Neural Network. The

third steganalysis was done using the a Fully Connected Neural Network (see Fig. 6). We denote this steganalysis, **FNN**, for Fully-Connected Neural Network.
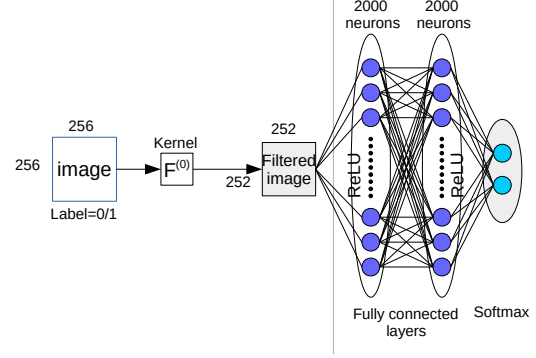


**Figure 6.** *A Fully Connected Neural Network (FNN).*

For each payload size, we conducted 10 tests where, for each test, the learning was done on 60 000 images randomly taken from the 80 000 images, such that the covers and stegos are always paired. The tests were performed on 10 000 images (5000 covers and 5000 associated stegos) randomly taken from the 20 000 remaining images. For the Ensemble Classifier, the decision threshold of each base learner was adjusted to minimize the total detection error under equal priors on the training set[9]. For CNN, and FNN, the sum of the errors was minimized on the training set, as a gradient descent:

$$min \sum_{i=1}^{i=N_{TR}} \sum_{l=0}^{l=1} (h_l(x_i) - y_i)^2, \tag{5}$$

with $N_{TR}$ being the number of training images, $h_0(.)$ (resp. $h_1(.)$), the CNN outputs giving a classification score for cover (resp. stego), $y_i \in \{0,1\}$ the label (0 for cover, and 1 for stego) for the $i^{th}$ input image $x_i$.

For both EC+RM, CNN, and FNN, the probability of error was computed and averaged over the 10 tests. The results are summarized in Table 1.

| | RM+EC | CNN | FNN |
|---|---|---|---|
| Max | 24.93% | 7.94% | 8.92% |
| Min | 24.21% | 7.01% | 8.44% |
| Variance | 0.14 | 0.12 | 0.16 |
| Average | 24.67% | 7.4% | 8.66% |

**Table 1. Steganalysis results under the clairvoyant scenario for RM+EC, CNN, and FNN**

Rich Models with the Ensemble Classifier gives a probability of error of 24.67%, whereas CNN gives a probability of error of

---

[7]$\alpha$=0.001, $\beta$=0.75, and *size*=9.

[8]Note that we also tested a CNN with a feature vector of dimension similar to SRM, by keeping the pooling steps, and 64 filters per layer, and we obtained a probability of error 10% better than with an EC with SRM.

[9]We used two different Ensemble Classifiers. The one given on the Binghamton website[10]; in that case the learning set only consists of 10 000 images, and the one written in C++ and proposed in the paper [11] with the features selection, and parallelization options; in that case the learning set consists of 60 000 images. For those two cases, the results were similar, and we only report here the results obtained with the Binghamton code.

only 7.4%, and FNN gives a probability of error of 8.75%. There is thus more than 17% improvement by using our CNN, and more than 15% with our FNN. This is a very impressive improvement considering the difficulty to grab percentages on the probability of error in steganalysis.

A first tentative explanation of the good behavior of the CNN has already been given in the previous Sections.

Our CNN generates better results than Rich Models associated with an Ensemble Classifier for the following reasons:

i) the shape of the CNN is well chosen,

ii) the learning process is done through a single global optimization,

iii) the filter kernels are optimized (which is not the case in Rich Models),

iv) there is a high number of filters (64 filters for the first layer) which enriches the diversity,

v) the second convolutional layer, which comes after what looks like a spatio-frequencial decomposition (obtained via the first layer), seems to act as the research of the presence of signals in all the spatio-frequential bands,

vi) we eliminated the pooling step (compared to the Qian *et al.* network) which was counter-productive because it was acting as a down-sampling, and thus leading to information loss.

Note that a less structured network such as FNN gives impressive results even though it is less efficient than CNN. Note that the unknowns number for CNN is around 259 million (only 29 824 unknowns for the two convolutions layers, and 259 million for the fully connected layers ), and for FNN is around 131 millions.

We also carried out an additional experiment that confirmed that the first two CNN layers play a strong role in classification. The experiment first consisted of "cutting" another "smaller" CNN (with a pooling step) that was previously learned, by only keeping the two convolution layers. The network was then nothing more than a *feature extractor*; in this experiment there were only 57600 features. Second, this feature extractor was used to extract, on the same learning database, one feature vector per image. Third, an Ensemble Classifier used those feature vectors to learn a model. The probability of error was computed and averaged over 10 tests. We observed that the Ensemble Classifier, that learned with the features coming from the "smaller" CNN, allowed to reduce the average probability of error by 0.4% w.r.t. the "smaller" CNN. This indicated that: i) feature extraction (i.e convolution steps) is the most interesting part of CNN, ii) the fully connected classifier part of CNN is not necessarily the best classifier. Note that this "smaller" CNN allowed us to obtain an improvement of 10% compared to RM+EC.

We would also like to comment on the adaptive scenario, i.e adaptive steganalysis [21], also named selection-channel-aware steganalysis [9]. When using adaptive steganalysis, the steganalyst uses an estimation of the modification probability of each pixel, as additional information for learning, in order to distinguish between cover and stego. In that scenario, authors from [9] report a detectability improvement of 1 to 4% with the Ensemble Classifier and SRM for detecting S-UNIWARD on the BossBase

database. At 0.4 bpp, the improvement was less than 2%. Those results are not comparable with ours since our images are smaller, but the 2% increment is really minor compared to the 17% CNN increment. Considering the most activated sections of the CNN, it seems that the CNN intrinsically guesses the selection channel.

### 3.4. Cover-Source Mismatch scenario

Our second test gave a very interesting results in the case of cover-source mismatch. The cover-source mismatch phenomenon occurs when the sources model, obtained during the learning step, differs from the sources that are used by the steganograph. From a geometrical point of view, we can explain this inconsistency problem by the fact that the cloud, describing the images used by the steganalyst, is not located at the same place as the cloud describing the images used by the steganograph. Only a few papers have assessed the cover-source mismatch in practice [12], and [22, 6]. Nevertheless, no satisfactory solution is currently available, even though there have been some attempts to understand the phenomenon [23, 24].

In this case, we subsampled the cropped BOSSBase five times, thus obtaining five different training sets. For each training set, we built a CNN, and then we applied the classification model on the LIRMMBase test database (The database is available at `http://www.lirmm.fr/~chaumont/LIRMMBase.html`). We report the average error probability over the five trials. Note that the cover-source mismatch was present since none of the BOSS-Base cameras are in LIRMMBase. Note also that we used the same script as that used for BOSSBase in order to generate the grey-level images. Moreover, the original images we used to build the LIRMMBase, were uncompressed and came from three known databases: Dresden, Raise, and Columbia. The average probability of error, for RM+EC, CNN, and FNN are given in Table 2.

|  | RM+EC | CNN | FNN |
|---|---|---|---|
| Max | 49.85% | 5.90% | 6.60% |
| Min | 47.20% | 4.00 % | 5.40% |
| Variance | 0.35 | 0.45 | 0.31 |
| Average | 48.29% | 5.16% | 5.89% |

**Table 2. Steganalysis results under the cover-source mismatch scenario for RM+EC, CNN, and FNN. The learning was done on BOSSBase and the tests on LIRMMBase.**

We observed that the cover-source mismatch issue seriously affected the performance of the RM+EC classifier. Its results were close to those obtained by a random classifier. In return, CNN showed incredible robustness to the mismatch phenomenon with a 5.16% probability of error. FNN gave similar results with a 5.96% probability of error. Other experiments on BO**W**SBase [25] also confirmed that CNN is robust to the cover-source mismatch phenomenon, whereas RM+EC is not. BO**W**SBase is nevertheless not a very practical database since the set of used cameras is unknown, and since some of the cameras have also been used in BO**S**SBase.

Surprisingly, the probability of error of CNN for LIRMMBase (5.16%) was lower than that from the cropped BOSSBase (7.4%). This was because there are fewer texture images in LIRMMBase than in the cropped BOSSBase. The LIRMMBase is thus easier to steganalyze. The CNN invariance to cover-source

mismatch was so good that the mismatch phenomenon was no longer present, and we thus obtained steganalysis results that were more related to the content complexity of the data-base [26] (Textured image databases are harder to steganalysis than homogenous ones).

The robustness was probably due to the transformation done by the two first layers (because keeping those layers and then branching an EC gives results similar to those of the whole CNN). Thus the feature extractor due to the first two layers gives features that are robust to the cover-source mismatch. Those features are probably invariant which means that the feature representation obtained after the second layer is not sensitive to the different image statistics. Assuming that the role of the first layer is to decompose the noise signal (if the image is stego, this noise is the stego noise) in a spatio-frequential decomposition, and that the role of the second layer is to detect the presence of particular patterns in the spatio-frequential bands, then this could explain the invariance to the image content and thus the invariance to the cover model.

# 4.Further discussion

In order to demystify CNN, we already explained that the learning was equivalent to minimization of a function having many unknown parameters with a technique similar to gradient descent. In this Section, we make links with previous research on the topic.

An important step of the CNN process is convolution as detailed in Sections 2.2 and 3.3. Learning of the filter kernels is done through minimization of the classification error using the back-propagation procedure. This is thus a simple optimization of filter kernels. This strategy already shown its efficiency in [27]. In that paper, some of the filter kernel values, which were used for computing the feature vector, were obtained through optimization with the downhill simplex algorithm. The objective was to minimize the probability of error given by an Ensemble Classifier. The learning achieved in the convolution layers of a CNN shares the same idea that leads to customized kernels, well suited for steganalysis purpose.

When looking more precisely at the first layer, the kernel seems to act as a multi-band filtering (see Figure 2, where we can see the 64 kernels of size $7 \times 7$ of our CNN). Some recent articles use such spatio-frequential decomposition (or a projection as in PSRM [28]) in order to compute Rich Models using Gabor filters [29] or DCT filters (DCTR features) [30]. Those filters are used to define projections that will then be used for computing a histogram leading to a feature vector.

When looking more precisely at the second convolution layer, which applies a very unusual convolution approach, nothing similar could be found in recent papers dealing with feature extraction, such as histogram computation [3, 28], non-uniform quantization [31], feature selection [11], dimension reduction [32], etc. Nevertheless something important is occurring in this second layer which allows us to obtain features unsensitive to the cover-source mismatch. By looking to the Equation 3, the role of this second layer looks like searching patterns in multiple bands. The sum of the convolutions from Equation 3 is a way to accumulate presence clues of a searched signal in all the subands. The second layer then outputs a set of maps giving some clues to a signal presence.

To close the discussion, we should also add that the last treatment of each layer is a normalization step, and that this type of processing can also be found in papers such as [33] or [10]. This normalization is done in order to obtain, comparable output values for each neuron. We should also mention that further analysis on the activation function is requiered to completely understand its impact. The activation introduced non-linearity, which is also the case in the Ensemble Classifier through the majority vote [5], or in Rich Models with the Min-Max features [3].

# 5.Conclusion

In this article, we propose to pursue the study of CNNs for steganalysis. We tested more than 40 CNNs and found the right parameters for the steganalysis domain. Instead of using a very deep network, such as that proposed by Qian *et al.*, our experiments led us to use a network that is in height, and only consists two convolutional layers. We replaced the unconventional Gaussian activation function by a more classical ReLU activation function, we suppressed the pooling step that was acting as a down-sampling, and we pre-processed the images by applying high pass filtering before feeding the CNN.

We evaluated CNNs in two different scenarios. The first set of tests was done with the clairvoyant scenario. We used the cropped BOSSBase database with embedding with S-UNIWARD at 0.4 bpp. Compared to the state-of-the-art approach, i.e. the Ensemble Classifier with SRM features, CNN and FNN reduce the classification error by a three fold.

The second set of tests was done with the cover-source mismatch scenario. We used the BOSSBase with S-UNIWARD embedding at 0.4 bpp for learning, and tests were carried out on the public LIRMMBase database. The cover-source mismatch was fully achieved since the cameras were different from one base to another. The conventional method (RM+EC) totally failed to detect the use of steganography in LIRMMBase since the classification error was 48.29% i.e almost a random classification. Conversely, CNN exhibited natural invariance to the cover-source mismatch with a classification error of 5.16%.

Our future studies will concentrate on improving some of the network parameters, and on gaining insight into the network behavior. Moreover, further experiments have to be done with different payload sizes, and different algorithms.

# References

[1] J. Fridrich, J. Kodovský, V. Holub, and M. Goljan, "Breaking HUGO - The Process Discovery," in *Proceedings of the 13th International Conference on Information Hiding, IH'2011*, Prague, Czech Republic, May 2011, vol. 6958 of *Lecture Notes in Computer Science*, pp. 85–101, Springer.

[2] J. Kodovský and J. Fridrich, "On Completeness of Feature Spaces in Blind Steganalysis," in *Proceedings of the 10th ACM Workshop on Multimedia and Security, MM&Sec'2008*, Oxford, United Kingdom, Sept. 2008, pp. 123–132.

[3] J. Fridrich and J. Kodovský, "Rich Models for Steganalysis of Digital Images," *IEEE Transactions on Information Forensics and Security, TIFS*, vol. 7, no. 3, pp. 868–882, June 2012.

[4] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[5] J. Kodovský, J. Fridrich, and V. Holub, "Ensemble Classifiers for Steganalysis of Digital Media," *IEEE Transactions on Information Forensics and Security, TIFS*, vol. 7, no. 2, pp. 432–444, 2012.

[6] I. Lubenko and A. D. Ker, "Steganalysis with Mismatched Covers: Do Simple Classifiers Help?," in *Proceedings of the 14th ACM multimedia and Security Workshop, MM&Sec'2012*, Coventry, United Kingdom, Sept. 2012, pp. 11–18.

[7] A. D. Ker, P. Bas, R. Böhme, R. Cogranne, S. Craver, T. Filler, J. Fridrich, and T. 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*, Montpellier, France, June 2013, pp. 45–58, ACM.

[8] T. Pevný, "Detecting Messages of Unknown Length," in *Proceedings of SPIE Media Watermarking, Security, and Forensics, Part of IS&T/SPIE 21th Annual Symposium on Electronic Imaging, SPIE'2011*, San Francisco, California, USA, Feb. 2011, vol. 7880.

[9] T. Denemark, V. Sedighi, V. Holub, R. Cogranne, and J. Fridrich, "Selection-Channel-Aware Rich Model for Steganalysis of Digital Images," in *Proceedings of the IEEE International Workshop on Information Forensics and Security, WIFS'2014*, Atlanta, GA, Dec. 2014, pp. 48–53.

[10] R. Cogranne, T. Denemark, and J. 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*, Atlanta, GA, Dec. 2014, pp. 167–172.

[11] 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*, Lake Buena Vista (suburb of Orlando), Florida, USA, Sept. 2012, pp. 1133–1136.

[12] J. Pasquet, S. Bringay, and M. Chaumont, "Steganalysis with Cover-Source Mismatch and a Small Learning Database," in *Proceedings of the 22nd European Signal Processing Conference 2014, EUSIPCO'2014*, Lisbon, Portugal, Sept. 2014, pp. 2425–2429.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25, NIPS'2012*, F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, Eds., pp. 1097–1105. Curran Associates, Inc., 2012.

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

[15] T. Pevný, T. Filler, and P. Bas, "Using High-Dimensional Image Models to Perform Highly Undetectable Steganography," in *Proceedings of the 12th International Conference on Information Hiding, IH'2010*, Calgary, Alberta, Canada, June 2010, vol. 6387 of *Lecture Notes in Computer Science*, pp. 161–177, Springer.

[16] V. Holub and J. Fridrich, "Designing Steganographic Distortion Using Directional Filters," in *Proceedings of the IEEE International Workshop on Information Forensics and Security, WIFS'2012*, Tenerife, Spain, Dec. 2012, pp. 234–239.

[17] V. Holub, J. Fridrich, and T. Denemark, "Universal Distortion Function for Steganography in an Arbitrary Domain," *EURASIP Journal on Information Security, JIS*, vol. 2014, no. 1, 2014.

[18] P. Bas, T. Filler, and T. Pevný, "'Break Our Steganographic System': The Ins and Outs of Organizing BOSS," in *Proceedings of the 13th International Conference on Information Hiding, IH'2011*, Prague, Czech Republic, May 2011, vol. 6958 of *Lecture Notes in Computer Science*, pp. 59–70, Springer.

[19] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, July 2006.

[20] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transaction on Pattern Analysis and Machine Intelligence, PAMI*, vol. 35, no. 8, pp. 1798–1828, 2013.

[21] Weixuan Tang, Haodong Li, Weiqi Luo, and Jiwu Huang, "Adaptive Steganalysis Against WOW Embedding Algorithm," in *Proceedings of the 2nd ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec'2014*, Salzburg, Austria, 2014, pp. 91–96.

[22] I. Lubenko and A. D. Ker, "Going from Small to Large Data in Steganalysis," in *Proceedings of SPIE Media Watermarking, Security, and Forensics III, Part of IS&T/SPIE 22th Annual Symposium on Electronic Imaging, SPIE'2012*, San Francisco, California, USA, Feb. 2012, vol. 8303.

[23] A. D. Ker and T. Pevný, "A Mishmash of Methods for Mitigating the Model Mismatch Mess," in *Proceedings of SPIE Media Watermarking, Security, and Forensics, Part of IS&T/SPIE 24th Annual Symposium on Electronic Imaging, SPIE'2014*, San Francisco, California, USA, Feb. 2014, vol. 9028, pp. 90280I–90280I–15.

[24] J. Kodovský, V. Sedighi, and J. Fridrich, "Study of Cover Source Mismatch in Steganalysis and Ways to Mitigate its Impact," in *Proceedings of SPIE Media Watermarking, Security, and Forensics, Part of IS&T/SPIE 24th Annual Symposium on Electronic Imaging, SPIE'2014*, San Francisco, California, USA, Feb. 2014, vol. 9028, pp. 90280J–90280J–12.

[25] P. Bas and T. 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/.

[26] G. Cancelli, G. J. Doërr, M. Barni, and I. J. Cox, "A Comparative Study of +/-1 Steganalyzers," in *Proceedings of the IEEE 10th Workshop on Multimedia Signal Processing, MMSP'2008*, 2008, pp. 791–796.

[27] V. Holub and J. 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*, San Francisco, California, USA, Feb. 2012, vol. 8303, pp. 830309–830309–13.

[28] V. Holub, J. Fridrich, and T. Denemark, "Random Projections of Residuals as an Alternative to Co-occurrences in Steganalysis," in *Proceedings of SPIE Media Watermarking, Security, and Forensics, Part of IS&T/SPIE Annual Symposium on Electronic Imaging, SPIE'2013*, San Francisco, California, USA, Feb. 2013, vol. 8665, pp. 86650L–86650L–11.

[29] 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'20015*, Portland, Oregon, USA, June 2015, pp. 15–23.

[30] V. Holub and J. Fridrich, "Low-Complexity Features for JPEG Steganalysis Using Undecimated DCT," *IEEE Transactions on Information Forensics and Security, TIFS*, vol. 10, no. 2, pp. 219–228, Feb 2015.

[31] T. 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*, San Francisco, California, USA, Feb. 2012, vol. 8303, pp. 83030B–83030B–13.

[32] T. Pevný and A. 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*, San Francisco, California, USA, Feb. 2013, vol. 8665, pp. 86650M–86650M–15.

[33] S. Kouider, M. Chaumont, and W. Puech, "Adaptive Steganography by Oracle (ASO)," in *Proceedings of IEEE International Conference on Multimedia and Expo, ICME'2013*, San Jose, California, USA, July 2013, pp. 1–6.

## Author Biography

*Lionel PIBRE received his Master's degree in Computer Science in 2015 from University of Montpellier. His Master's degree allowed him to study data mining, visualization, machine learning and NLP. He made his internship of Master's degree in the LIRMM laboratory in Montpellier. During this internship, he worked on steganalysis by Deep Learning. He is currently doing his Ph.D. thesis in the LIRMM laboratory. His work areas are multimedia security (steganography / steganalysis), segmentation and tracking in images.*



*Jérôme PASQUET received a master degree in computer science from the university of Montpellier II, France, in 2013. He is currently working toward the PhD degree in LIRMM (Montpellier Laboratory of Informatics, Robotics and Microelectronics). His research interests are urban objects detection and segmentation in aerial photography.*



*Dino IENCO received his PhD in Computer Science in 2010 from University of Torino. From 2011 he is researcher at at the Irstea Institute, Montpellier, France. His main topics of research involved clustering algorithm for textual information, social network analysis, data mining approaches for biological data, supervised and semi-supervised methods for multimedia data (image and document analysis) and graph mining. He is actively working in the field of spatio-temporal data with a major emphasis on remote sensing analysis. In 2009, he visited the Yahoo! Research Lab in Barcelona (Spain) to work on information propagation. He co-authored more than 40 papers in major international conferences (SDM, ECML/PKDD, EMNLP, ECIR, etc.), and journals (DAMI, ACM TKDD, Pattern Recognition, JIIS, etc.).*
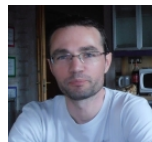


*Marc CHAUMONT received his Engineer Diploma in Computer Sciences at the INSA of Rennes, France in 1999, his Ph.D. at the IRISA Rennes in 2003, and his HDR ("Habilitation à Diriger des Recherches") at the University of Montpellier in 2013. Since September 2005, he is an Assistant Professor in the LIRMM laboratory of Montpellier and the University of Nmes. His research areas are multimedia security (steganography, watermarking, digital forensics, video & image compression) and segmentation & tracking in images and videos. He is member of the TC of IEEE SPS - Information Forensics and Security for the period 2015-2017. He was program chair of ACM IH&MMSec'2013. He is reviewer for more than 20 journals (IEEE TIFS, IS&T JEI, ...) and for more than 10 conferences (EI MWSF, IEEE WIFS, ACM IH&MMSec, IEEE ICIP, ...).*