Pooled Steganalysis in JPEG: how to deal with the spreading strategy?

Ahmad Zakaria, Marc Chaumont, Gérard Subsol

▶ **To cite this version:**

Ahmad Zakaria, Marc Chaumont, Gérard Subsol. Pooled Steganalysis in JPEG: how to deal with the spreading strategy?. WIFS 2019 - 11th IEEE International Workshop on Information Forensics and Security, Dec 2019, Delft, Netherlands. 10.1109/WIFS47025.2019.9035096 . hal-02301698

HAL Id: hal-02301698
https://hal.science/hal-02301698

Submitted on 30 Sep 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.

Pooled Steganalysis in JPEG: how to deal with the spreading strategy?

Ahmad ZAKARIA
LIRMM, Univ Montpellier,
CNRS,
France.
Email: ahmad.zakaria@lirmm.fr

Marc CHAUMONT
LIRMM, Univ Montpellier,
CNRS, Univ Nîmes,
France.
Email: marc.chaumont@lirmm.fr

Gérard SUBSOL
LIRMM, Univ Montpellier,
CNRS,
France.
Email: gerard.subsol@lirmm.fr

***Abstract*—In image pooled steganalysis, a steganalyst, Eve, aims to detect if *a set* of images sent by a steganographer, Alice, to a receiver, Bob, contains a hidden message. We can reasonably assess that the steganalyst does not know the strategy used to spread the payload across images. To the best of our knowledge, in this case, the most appropriate solution for pooled steganalysis is to use a Single-Image Detector (SID) to estimate/quantify if an image is cover or stego, and to average the scores obtained on the *set* of images.**

In such a scenario, where Eve does not know the spreading strategies, we experimentally show that if Eve can *discriminate* among few well-known spreading strategies, she can improve her steganalysis performances compared to a simple averaging or maximum pooled approach. Our *discriminative* approach allows obtaining steganalysis efficiencies comparable to those obtained by a clairvoyant, Eve, who knows the Alice spreading strategy. Another interesting observation is that DeLS spreading strategy behaves really better than all the other spreading strategies.

Those observations results in the experimentation with six different spreading strategies made on Jpeg images with J-UNIWARD, a state-of-the-art Single-Image-Detector, and a *discriminative* architecture that is invariant to the individual payload in each image, invariant to the size of the analyzed set of images, and build on a binary detector (for the pooling) that is able to deal with various spreading strategies.

I. Introduction

Steganography consists in altering a digital object (called *cover*), in an innocuously looking way, to hide (or embed) a message which allows secret message communication. The resulting altered object is named *stego*. The science of *detection* of the presence of this hidden data, given an innocuous object, is called *steganalysis*. In this paper, we focus on the steganalysis of digital images, which are the most studied cover objects. More precisely, we use images coded in JPEG format which is the most common one nowadays.

Steganography traditionally focused on embedding a message in one image at a time, but it is much more realistic for the steganographer to hide the message by *spreading* it over multiple images. This *spreading* is called *batch steganography* and is to be opposed to the *pooled steganalysis* where a *set* of images are analyzed in order to gather a set of clues, and thus to conclude to the presence/absence of a hidden message in the *set of images*. Batch steganography and pooled steganalysis topics were introduced in [1] and became one of the most challenging open problems in the field these last years [2].

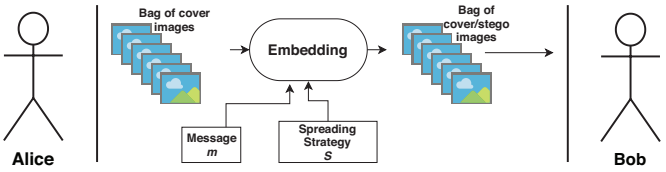

Fig. 1. A scheme that illustrates how the steganographer, Alice, spreads a message $\mathbf{m}$ in multiple covers using a strategy $s \in \mathcal{S}$.

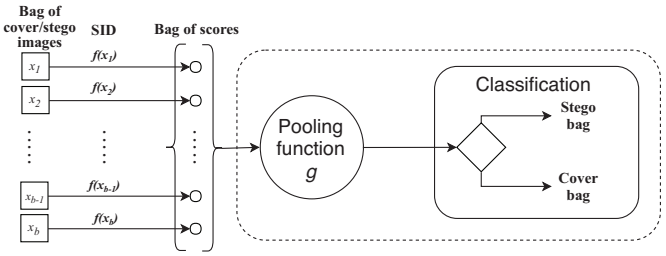

Fig. 2. A scheme that illustrates how the steganalyst, Eve, uses a pooling function g to aggregate evidence from multiple images in order to make a final decision about the presence/asbcence of a hidden message.

As presented in Figure 1, *batch steganography* consists in embedding a message $\mathbf{m} \in \{0,1\}^{|m|}$, of length $|m|$, in a bag of cover images by using a spreading strategy, $s \in \mathcal{S}$, from a set of possible spreading strategies $\mathcal{S}$.

As presented in Figure 2, modern *pooled steganalysis* consists to take a bag B of b images $\{x_1, ..., x_b\}$, which may be cover or stego, applies a Single Image Detector (SID), denoted by the function f, which can be based either on a binary or a quantitative algorithm, in order to get the set of SID scores $\{f(x_1), ..., f(x_b)\}$. Then the steganalyst aggregate these scores by using a pooling function $g : \mathbb{R}^b \mapsto \mathbb{R}$ in order to get a single output which allows to classify the bag as *cover* or *stego*.

The first attempts in order to *experimentally* evaluate batch steganography / pooled steganalysis has started in 2011 and 2012 [3], [4], with the paradigm of *multiple users* (multiple actors), and the research of outliers. Instead of making an individual and independent binary decision on which actor is guilty, the proposed algorithms rank the actors according to their guiltiness. Nevertheless, those algorithms does not work in a *single actor* case, which is a more general, and thus more

interesting working case.

Since 2015, three papers address the problem of only one actor and a pooling defined by the aggregation of clues (SID scores) computed on each image individually [5], [6], and [7].

In the article [5], Cogranne deals with the case where *Eve does not know the spreading strategy* used by Alice. In that case, Cogranne shows that the best pooling strategy consists of *averaging* the individual scores. In the article [6], Pevny and Nikolaev deal with the case where *Eve does know the spreading strategy* used by Alice. In that case, Pevny and Nikolaev observe that the knowledge of the strategy allows improving the steganalysis results. In the article [7], Cogranne also shows that if *Eve does know the spreading strategy* used by Alice, it allows to better aggregate the individual SID scores, and thus to obtain better steganalysis results. The tendency shared by those papers is that the optimal pooling function g applied on the SID scores depends on the steganographer's strategy used to spread the messages among multiple objects (this was also expressed in the seminal paper [1] in the case of non-adaptive embedding).

In our paper, we study the scenario where *Eve does not know* the spreading strategy used by Alice, and we propose to evaluate Eve's capability to obtain better results than those she would obtain by averaging the SID scores. To reach that goal, first, Eve *learns* well-known spreading strategies, and secondly, during the test time, we test Eve's ability to *discriminate* between a *cover* bag and a *stego* bag. During the test, Eve applies a weighted sum during pooling instead of an average. Our approach does not guess the spreading strategy used by Alice even if, thanks to the learning process, Eve is able to better *discriminate* between *cover* bag and *stego* bag, by distinguishing better the various statistics associated to each strategy. The reader must understand that compared to the previous papers and especially to [6], the question which is raised in our paper is first, about the ability for Eve to use some knowledge about the spreading strategies (without knowing exactly which one is used by Alice), and second, to do it in a more realist scenario (no knowledge about the individual payload sizes, no knowledge about the spreading strategy, and eventually no knowledge of the bag size).

In Section II we briefly describe a state-of-the-art list of spreading strategies and a we present a general pooled steganalysis architecture. In Section III we discuss more in detail the question raised in our paper. Finally, we present in Section V the results and discuss them. We conclude and give some perspectives in Section VI.

II. Batch spreading strategies and a general pooled steganalysis architecture

A. Batch spreading strategies ($\mathcal{S}$)

Note that the spreading in a bag is done at a given payload size which is expressed in bit per total coefficients (bptc). The bptc is thus the size of the message in bits divided per the total number of pixels (i.e. the ACs and DCs coefficients) of all the images in the bag. We have compiled the batch spreading strategies proposed in [6], [7], [8] in the following list:

1) **Greedy strategy:** the steganographer embeds the message into as few covers as possible. The steganographer randomly chooses a cover and embeds part of his message up to 1 bit per coefficients (bpc)[1], because the intrinsic security of the image is not taken into account in the greedy strategy. Eve repeats the embedding process with another randomly chosen cover until the whole message bits are embedded.
2) **Linear strategy:** the steganographer distributes the message evenly across all available covers, so the payload size, in bits, for each image is equal to the message length divided by b.
3) $Uses$-β **strategy:** the steganographer distributes the message evenly across a fraction β of available cover images. This strategy is equivalent to the greedy one for $\beta = \alpha$, and to the linear one for $\beta = 1$.
4) **Image Merging Sender (IMS) strategy:** the steganographer generates a unique image from all the b images from a bag B, and lets the embedding algorithm spread the payload all over this "big" image. More precisely a cost value is computed for each DCT coefficient of the image (the adaptive embedding algorithm define the way the cost map is computed), and then the embedding is obtained with STC [9] or the simulator [10].
5) **Detectability Limited Sender (DeLS) strategy:** the steganographer adopts a cover model and spreads payload over b images that communicates the required payload, so that each image from the bag contributes with the same value as the Kullback-Leibler (KL) divergence (deflection coefficient) based on MiPOD [11] cover model[2].
6) **Distortion Limited Sender (DiLS) strategy:** The steganographer spreads payload over images so that each image from the bag contributes with the same value of distortion.

B. General pooled steganalysis architecture

In this section we recall the general pooled steganalysis architecture that were proposed in [6] for evaluating a given pooled steganalysis facing a given spreading strategy. Figure 3 resumes the different steps applied by Eve when she analyzes a bag of images.

In the operational phase (i.e. when the general architecture is deployed), the pooled steganalysis algorithm takes as input a bag made of b images $\{x_1, ..., x_b\}$ which may be cover or stego, and computes for each image the SID score, which is a real positive number, through the f function (see Section I and III). It thus gives a bag of real numbers $\{f(x_1), ..., f(x_b)\}$. From this bag made of b values, a *Parzen window* (detailed below) is computed and lead to a histogram, noted $\mathbf{h}$, and made of p bins. Finally, the pooling function aggregates the p

[1]The choice of 1 bpc for the maximum payload makes the construction of the bag practically easier.

[2]The deflection coefficient is computed on dequantized rounded images.

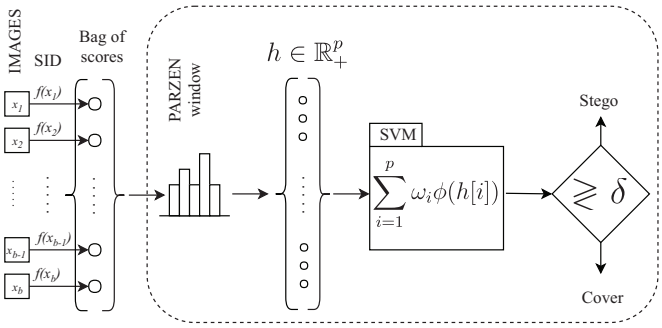

Fig. 3. The general pooled steganalysis architecture from [6]. ϕ is the re-description transformation function. δ is a threshold.

values of the histogram. The resulting weighted sum is then compared to a threshold, noted δ in order to decide if the bag is cover or stego.

Let us comment the Parzen window which is the most important ingredient of the architecture proposed in [6]. The bag of SID scores, i.e. the vector $\mathbf{z} = \{f(x_1), ..., f(x_b)\}$, is transferred into a histogram representation thanks to the estimation by Parzen window. Given the Gaussian kernel function $k : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ with $k(x, y) = exp(-\gamma||x - y||^2))$, the Parzen window computation is such that for a bag $\mathbf{z}$, the resulting histogram is:

$$\mathbf{h} = \left[\frac{1}{b} \sum_{f(x_i) \in z} k(f(x_i), c_1), \ ... \ , \ \frac{1}{b} \sum_{f(x_i) \in z} k(f(x_i), c_p) \right] \tag{1}$$

with $\{c_i\}_{i=1}^p$ a set of equally spaced real positive values belonging to the range $min_{x \in \mathcal{X}} f(x)$ and $max_{x \in \mathcal{X}} f(x)$, with $\mathcal{X}$ the images learning set. Each bin of the histogram $\mathbf{h}$, from Equation 1, is the result of the cumulative Gaussian distance between each component of $\mathbf{z}$ and a scalar from the set of predefined centers $\{c_i\}_{i=1}^p$.

Note that the histogram representation, $\mathbf{h}$, is of finite dimension p, whatever the dimension b of the bag, and that this representation is invariant to the sequential order in the bag.

Once the Parzen window is applied, the vector $\mathbf{h}$ of fixed dimension p is given to an SVM which pools the vector component in the re-description space for classification:

$$\sum_{i=1}^{p} \omega_i \ \phi(h[i]), \tag{2}$$

with ϕ the function redefining the feature space. Note that ϕ is, in practice, never computed because of the "kernel trick". We can see when looking to Equation 2, that the pooling function is a weighted sum where the weights ω_i are learnt during the SVM training. It is clearly more subtle to pool the set of features $\{\phi(h[i])\}_{i=1}^p$ of the bag than using the straightforward average or maximum.

III. Technical details

A. Assumptions and limits of the general pooled steganalysis architecture

In order to construct a general pooled steganalysis architecture, we should first choose a generic SID. It could be a quantitative detector such as [12], [6], [13], [14], or a detector outputting a score such as [15]. In this paper, we will use the quantitative detector described in [12].

To be the more realist possible, this SID should be invariant to image size (see some preliminary work in [16]) for a given detectability and should be robust to cover-source-mismatch (CSM) and to the stego-mismatch. The CSM problem is described in [17]; a holistic solution is proposed in [18], and an atomistic solution proposed in [19]. An example of work related to the stego-mismatch can be found in [20]. In this paper, we will assume that the size of the images is fixed and that there is no cover-source mismatch, neither stego-mismatch.

A general pooled steganalysis architecture must also be able to process a bag of any number of images. This is the case of the algorithms proposed in [5] or [6] even if in those papers, experiments are performed by using bags of only one size. On the contrary, this is not possible in the algorithm described in [7] where the hypothesis is that the steganalyst knows the number of images in the bag to analyze. In this paper, we propose an architecture dealing with any number of images in the bag, but in the experiments, we have trained our model on bags of fixed sizes. We postpone the experiment with bags of various size for future work.

Lastly, such architecture must be able to process a bag of any payload size, which is not done in any of the papers [5], [6], [7]. In our paper, we make the same assumption as [7] where the steganalyst learns at a fixed mean payload for each bag, and we postpone the experiments with any payload size in the bag for future work.

B. Single Image Detector (SID)

We choose as Single Image Detector (SID), which is referred in this paper as the function f, the feature-based Quantitative steganalysis algorithm proposed in [12], applied on the 17,000-dimensional JPEG domain Rich Model — the Gabor features residuals (GFR) [21]. The Quantitative steganalysis algorithm is a machine learning regression framework that assembles, via the process of gradient boosting, a large number of simpler base learners built on random subspaces of the original high-dimensional feature space.

C. Pooling functions

As explained in Section 3, the general pooled steganalysis architecture provided in [6] produces a feature space which is represented by a Parzen histogram, and proposes to pool the values of this histogram thanks to a linear SVM classifier. In the paper [6], only *one pooling* function (the SVM) is learnt for *one spreading strategy* since the study was on the comparison with the historical average and maximum pooling functions, depending on the embedded payload size in a

bag. Additionally, the experiment done in [6] only uses old spreading strategies (greedy and linear; see Section II-A).

In our study, we look at the behaviour of the architecture when it has learned to recognize *various spreading strategy*. Our approach is thus a pooling function which is able to face multiple spreading strategies. The experiment objective is to show that even if Eve does not have any information on the spreading strategy used by Alice, she can obtain better steganalysis results than when using a simple average or maximum, and she can be close to the results that she would obtain if she was clairvoyant i.e. if she knows the spreading strategy. The difference compared with the paper [6] is in the addressed question, and our paper is thus in the natural continuity of the three state-of-the-art papers [6], [7], [8]. The experiments are in agreement with various spreading strategies (the modern and recent ones) and with state-of-the-art two-step machine learning in order to build the Rich Model and the SID.

In order to study the efficiency of our *discriminative* pooling function in order to *discriminate* s among a set of strategies $\mathcal{S}$, we defined various pooling functions:

1) g_{disc}**:** This function is our *discriminative* pooling function and we train it on the Parzen histograms of all the strategies from $\mathcal{S}$. During the test, only one spreading strategy s will be tested at a time. g_{disc} is obtained through the learning of the various patterns from all the strategies, and the minimization of the classification error between a cover bag and a stego bag (whatever the spreading strategy) during the SVM learning. Thanks to the Parzen representation, the SVM re-description space, and the weighted sum, the general architecture learns with different spreading strategies, and it should be able to classify better than applying an average or a maximum.
2) g_{clair}**:** This function is the *clairvoyant* one. The training and the test are done with the knowledge of the used spreading strategy. The pooling function is obtained thanks to the use of an SVM, similarly as the g_{disc} function.
3) g_{max}**:** This function is only a maximum applied on a Parzen histogram. The threshold τ_{max} is obtained by minimizing, on all the strategies, the total classification error probability under equal priors $P_e = \frac{1}{2}(P_{fa}+P_{md})$, where P_{fa} and P_{md} are the false-alarm and missed-detection probabilities.
4) g_{mean}**:** This function is only an average applied on a Parzen histogram. The threshold τ_{mean} is obtained by minimizing, on all the strategies, the total classification error probability under equal priors $P_e = \frac{1}{2}(P_{fa}+P_{md})$, where P_{fa} and P_{md} are the false-alarm and missed-detection probabilities.

IV. Experimental evaluation

In this section, we compare g_{disc} to g_{mean}, g_{max} and g_{clair}.

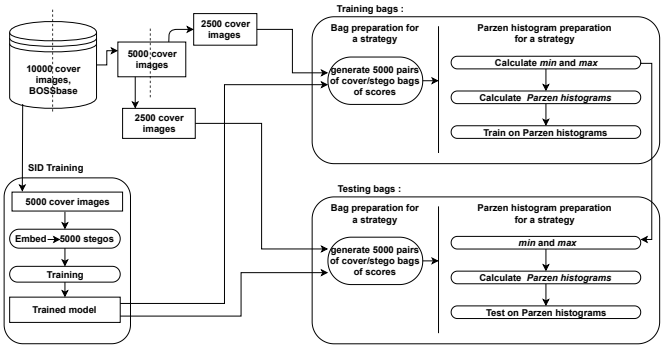

Fig. 4. Protocol for the data-sets creation, the learning and the test.

A. Data preparation

Our image database is built from the BOSSbase 1.01 [22]. We convert those 10 000 512×512 grey-scale **spatial** images into JPEG images, using the MATLAB's command *imwrite*, with quality factors 75.

The 10 000 cover JPEG images are split in two equal sizes sets as shown in Figure 4. The first set (5 000 cover images) is used for the learning of the quantitative SID. The second set (5 000 cover images) is used to create bags, learn the pooling functions, and test the various pooling.

Note that each time an embedding in an image is done, it is done with the J-UNIWARD scheme.

Also note that each time a SID is used (in the first or second test, during the learning and also during the test, on a cover or on a stego whatever the payload size), for a given input image, a feature vector *Gabor Features Residuals* (GFR) [21] of dimension 17 000 is first extracted. This feature vector is then cleaned from NaN values (it occurs when the feature values are constant over images) and from constant values, to obtain a 16 750-dimensional feature vectors. Finally, we normalize this vector using the algorithm proposed in [23].

B. SID learning

In order to train the quantitative Single Image Detector, we generate stegos such that there are 5 000 feature vectors for cover images and 5 000 feature vectors for *each* payload size whose range is fixed to $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ bpc[3] which gives a total of 55 000 features vectors. Indeed, while training the quantitative SID, to avoid any bias in the results, we use 5 000 covers and 50 000 stegos such that the cover and each one of the ten different payload sizes are equally distributed. This way the resulted SID model will guarantee a fair scoring between each payload (payload 0 for covers).

C. Bags preparation

The remaining 5 000 cover images are used for batch steganography i.e. bags preparation. More precisely 2 500 cover images are used for the learning, and the remaining 2 500 cover images will be used for the test.

[3]We adapted the J-UNIWARD algorithm to insert an amount of data measured in bpc instead of bpnzAC.

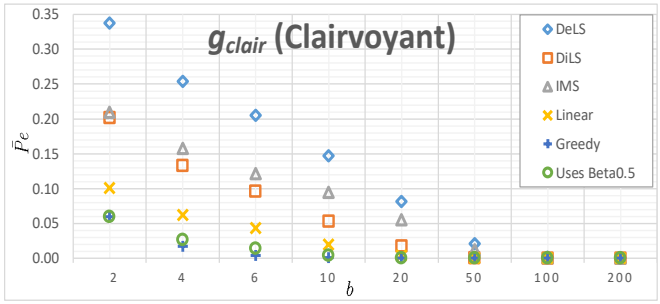

Fig. 5. Spreading strategies comparison in the *clairvoyant* case. Average probability of error under equal prior, $\overline{Pe}$, as a function of pooling bag size $b \in \mathcal{B}$ for an average payload 0.1 bptc for g_{clair} pooling function.

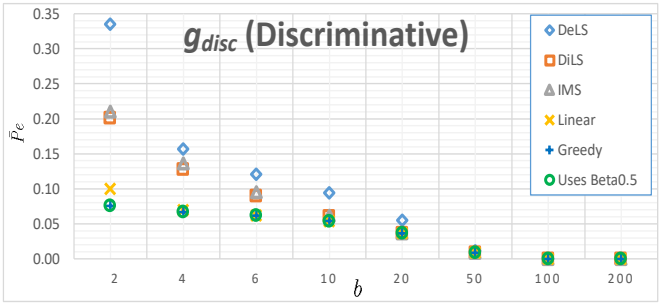

Fig. 6. Spreading strategies comparison in the *discriminative* case. Average probability of error under equal prior, $\overline{Pe}$, as a function of pooling bag size $b \in \mathcal{B}$ for an average payload 0.1 bptc for g_{disc} pooling function learnt over all the strategies.

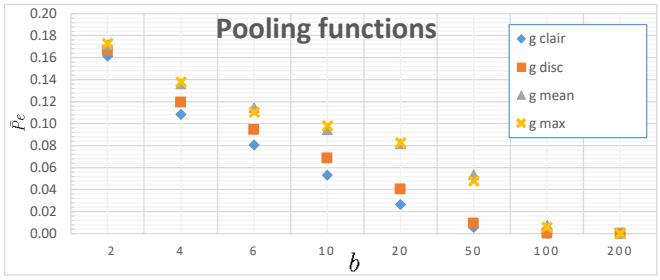

Fig. 7. Pooling steganalysis comparison. Average probability of error under equal prior, $\overline{Pe}$, as a function of pooling bag size $b \in \mathcal{B}$ for an average payload 0.1 bptc. The average $\overline{Pe}$ is computed by testing each spreading strategy.

Given a bag size $b \in \mathcal{B} = \{2, 4, 6, 10, 20, 50, 100, 200\}$, and the set of spreading strategies IMS, $DeLS$, $DiLS$, $Greedy$, $Linear$ and $Uses\text{-}\beta$ (β is fixed to 0.5), we generate a total of 30 000 stego bags plus 5 000 cover bags, with inside each bag a bit-rate of $\bar{R}$ = 0.1 bptc. Whatever the pooling function, a set of 5 000 pairs of cover/stego bags (5 000 cover bags and 5 000 stego bags) is used. For each bag, the Rich Model, and the SID score is computed leading to a vector of SID scores. All the process is illustrated in Figure 4.

D. g_{disc} learning

Given the 35 000 bags obtained for a bag size $b \in \mathcal{B}$ and all the spreading strategies (IMS, $DeLS$, $DiLS$, $Greedy$, $Linear$ and $Uses\text{-}\beta$), the minimum and the maximum of the SID scores are computed, which allows to define $p = 100$ centers, equally spaced in the range $[min_{x\in\mathcal{X}} f(x), max_{x\in\mathcal{X}} f(x)]$, as in [6]. A Parzen histogram $\mathbf{h}$ can then be computed for each of the 35 000 bags. The pooling function g_{disc}, uses for the learning, 5 000 cover bags (i.e 5 000 Parzen vectors $\mathbf{h}$) and their corresponding 5 000 stego bags (i.e 5 000 Parzen vectors $\mathbf{h}$) equally distributed on the six spreading strategies. Note that g_{clair} uses 5 000 cover bags and their corresponding 5 000 stego bags for one strategy.

The classifier, used for learning g_{disc}, is a SVM with a linear kernel. We use the SVM package from the free software machine learning library for the Python programming language Scikit-Learn [24]. The parameters are set to default, but the value of the kernel is set to 'linear'.

V. Results

As shown in Figure 4, tests are done by using 2 500 cover image never seen, which allow to form a set of 5 000 pairs cover/stego bags for a fixed size b for an spreading strategy (IMS, $DeLS$, $DiLS$, $Greedy$, $Linear$ and $Uses\text{-}\beta$). In order to generate a stego bag, a set of b covers is randomly picked among the 2500 images, and the spreading strategy is then executed. The average probability of error, obtained by each pooling function, over 10 runs done each time with a different learning set (and testing set) of 5000 pairs of cover/stego bags, is then reported.

Figure 5 reports the results obtained with the *clairvoyant* steganalysis scenario i.e. with g_{clair}. One can notice that the DeLS is the best and it outperforms the IMS, which was more competitive to $DeLS$ in [7], while the Greedy strategy is the worst. One can cluster the strategies into 3 groups: the $Greedy$ and $Uses\text{-}\beta$ which are highly detectable with a detectability which starts to coincide for bag sizes ≥ 10 with $\overline{Pe} \approx 0$, the strategies $DeLS$, IMS and $DiLS$ which are the more secure ones, and the $Linear$ strategy which falls between the two other groups. The strategies $DeLS$ and IMS becomes totally detectable at $b = 100$ with $\overline{Pe} \approx 0$.

In Figure 6, we report the detection of each spreading strategy with the *discriminative* g_{disc} pooling function. The best strategies are in the descending order, in the sense of its ability to resist the pooling function that tries to *discriminate* it, $DeLS$, IMS, $DiLS$, $Linear$, $Uses\text{-}\beta$, $Greedy$. $DeLS$ is again performing well in this *non-clairvoyant* approach, and it remains resistant until $b = 100$ where its average probability of error $\overline{Pe}$ start to coincide with those of the other strategies and becomes ≈ 0. $DeLS$ and IMS are more detected by the discriminative g_{disc} than the clairvoyant g_{clair}. Looking at the histograms of the SVM scores, and to the ROC curves (not shown in this paper), we observe indeed a higher separation with g_{disc}. Looking to Figure 6, we also observe a smaller gap between all the $\overline{Pe}$ of each strategy, compared to Figure 5. Those behaviour are probably because the optimization (learning of the SVM) try to minimize the prediction error fairly for each of the strategies.

Finally, Figure 7 provides a useful insight regarding the accuracy of the pooling methods. This figure shows the *average* probability of error under equal prior, $\overline{Pe}$, as a function of pooling bag size $b \in \mathcal{B}$, with an average payload size $\bar{R}$ = 0.1

bptc for each pooling function over all the strategies. We note that the g_{disc} pooling function outperforms g_{mean} and g_{max} functions with average difference of $\overline{Pe} \approx 2\%$ and is closer to g_{clair} with an average difference of $\overline{Pe} \approx 0.8\%$[4]. From figure 7, we could observe that g_{disc} and g_{clair} are more stable than g_{mean} and g_{max}. This is in agreement with the hypothesis of our paper which is that a *discriminative* pooling function allows obtaining better detection results compared to the mean and max pooling function.

VI. Conclusion

In this paper, we studied the problem of content-adaptive batch steganography and pooled steganalysis for a steganalyst unaware of the payload-spreading strategy and equipped with a single-image detector trained as a quantitative classifier.

We studied the ability of the steganalyst to *discriminate* the spreading strategy, thanks to a pooling function that is able to recognize various stego patterns, and then able to pool the SID scores much cleverly than applying a simple average or maximum. Empirical results made with six different spreading strategies and a state-of-the-art Single Image Detector confirms that our *discriminative* pooling function can improve the accuracy of the pooled steganalysis. Our pooling function gets results close to a $clairvoyant$ steganalyst which is assumed to know the spreading strategy.

This conclusion opens the door to future studies related to performance of a steganalyst that would not be aware of the bag's payload, to the performance of the steganalyst in the case of a spreading strategy never seen before, to the performance of an all integrated solution using Deep Learning, to the extension to a game-theory or practical GAN simulation, etc.

Acknowledgment

We would like to thank Dr Rémi Cogranne for discussions and for providing the MATLAB code of the DiLS, DeLS and IMS. We would like to thank the MESO@LR computing center, Montpellier, France, for providing a huge amount of calculation resources. We would like to thank the French Direction Générale de l'Armement (DGA) for its support through the Alaska project ANR (ANR-18-ASTR-0009). We would like to thank the Union of Municipalities of Jered Al-Kaytee, Akkar, Lebanon, for its scholarship support.

References

[1] A. D. Ker, "Batch steganography and pooled steganalysis," in *Information Hiding, 8th International Workshop, IH'06, Alexandria, VA, USA, July 10-12, 2006*, 2006, pp. 265–281.

[2] A. D. Ker, P. Bas, R. Böhme, R. Cogranne, S. Craver, T. Filler, J. J. Fridrich, and T. Pevný, "Moving steganography and steganalysis from the laboratory into the real world," in *ACM Information Hiding and Multimedia Security Workshop, IH&MMSec '13, Montpellier, France, June 17-19, 2013*, 2013, pp. 45–58.

[3] A. D. Ker and T. Pevný, "A new paradigm for steganalysis via clustering," in *Media Forensics and Security III, San Francisco Airport, CA, USA, January 24-26, 2011, Proceedings*, 2011, p. 78800U.

[4] ——, "Batch steganography in the real world," in *Multimedia and Security Workshop, MM&Sec 2012, Coventry, United Kingdom, September 6-7, 2012*, 2012, pp. 1–10.

[5] R. Cogranne, "A sequential method for online steganalysis," in *2015 IEEE International Workshop on Information Forensics and Security, WIFS 2015, Roma, Italy, November 16-19, 2015*, 2015, pp. 1–6.

[6] T. Pevný and I. Nikolaev, "Optimizing pooling function for pooled steganalysis," in *2015 IEEE International Workshop on Information Forensics and Security, WIFS 2015, Roma, Italy, November 16-19, 2015*, 2015, pp. 1–6.

[7] R. Cogranne, V. Sedighi, and J. J. Fridrich, "Practical strategies for content-adaptive batch steganography and pooled steganalysis," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, 2017, pp. 2122–2126.

[8] A. D. Ker and T. Pevný, "The steganographer is the outlier: Realistic large-scale steganalysis," *IEEE Trans. Information Forensics and Security*, vol. 9, no. 9, pp. 1424–1435, 2014.

[9] T. Filler, J. Judas, and J. Fridrich, "Minimizing additive distortion in steganography using syndrome-trellis codes," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 920–935, Sep. 2011.

[10] J. J. Fridrich, "Minimizing the embedding impact in steganography," in *Proceedings of the 8th workshop on Multimedia & Security, MM&Sec 2006, Geneva, Switzerland, September 26-27*, 2006, pp. 2–10.

[11] V. Sedighi, R. Cogranne, and J. J. Fridrich, "Content-adaptive steganography by minimizing statistical detectability," *IEEE Trans. Information Forensics and Security*, vol. 11, no. 2, pp. 221–234, 2016.

[12] J. Kodovský and J. J. Fridrich, "Quantitative steganalysis using rich models," in *Media Watermarking, Security, and Forensics 2013, Burlingame, CA, USA, February 5-7, 2013, Proceedings*, 2013, p. 86650O.

[13] A. Zakaria, M. Chaumont, and G. Subsol, "Quantitative and binary steganalysis in JPEG: A comparative study," in *26th European Signal Processing Conference, EUSIPCO 2018, Roma, Italy, September 3-7, 2018*, 2018, pp. 1422–1426.

[14] M. Chen, M. Boroumand, and J. J. Fridrich, "Deep learning regressors for quantitative steganalysis," in *Media Watermarking, Security, and Forensics 2018, Burlingame, CA, USA, 28 January 2018 - 1 February 2018*, 2018.

[15] R. Cogranne and J. J. Fridrich, "Modeling and extending the ensemble classifier for steganalysis of digital images using hypothesis testing theory," *IEEE Trans. Information Forensics and Security*, vol. 10, no. 12, pp. 2627–2642, 2015.

[16] C. F. Tsang and J. J. Fridrich, "Steganalyzing images of arbitrary size with cnns," in *Media Watermarking, Security, and Forensics 2018, Burlingame, CA, USA, 28 January 2018 - 1 February 2018*, 2018.

[17] G. Cancelli, G. Doërr, M. Barni, and I. J. Cox, "A comparative study of±steganalyzers," in *Multimedia Signal Processing, 2008 IEEE 10th Workshop on*. IEEE, 2008, pp. 791–796.

[18] I. Lubenko and A. D. Ker, "Going from small to large data in steganalysis," in *Media Watermarking, Security, and Forensics 2012, Burlingame, CA, USA, January 22, 2012, Proceedings*, 2012, p. 83030M.

[19] J. Pasquet, S. Bringay, and M. Chaumont, "Steganalysis with cover-source mismatch and a small learning database," in *22nd European Signal Processing Conference, EUSIPCO 2014, Lisbon, Portugal, September 1-5, 2014*, 2014, pp. 2425–2429.

[20] Y. Yousfi, J. Fridrich, J. Butora, and Q. Giboulot, "Breaking ALASKA: Color Separation for Steganalysis in JPEG Domain," in *Proceedings of the 7th ACM Workshop on Information Hiding and Multimedia Security*, ser. IH&MMSec'19, Paris, France, Jul. 2019.

[21] X. Song, F. Liu, C. Yang, X. Luo, and Y. Zhang, "Steganalysis of adaptive JPEG steganography using 2d gabor filters," in *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec 2015, Portland, OR, USA, June 17 - 19, 2015*, 2015, pp. 15–23.

[22] P. Bas, T. Filler, and T. Pevný, ""break our steganographic system": The ins and outs of organizing BOSS," in *Information Hiding - 13th International Conference, IH 2011, Prague, Czech Republic, May 18-20, 2011, Revised Selected Papers*, 2011, pp. 59–70.

[23] M. Boroumand and J. J. Fridrich, "Nonlinear feature normalization in steganalysis," in *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec 2017, Philadelphia, PA, USA, June 20-22, 2017*, 2017, pp. 45–54.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4]The variance on $\overline{Pe}$ for each pooling is around $\times 10^{-6}$.