Effect of Acquisition Noise Outliers on Steganalysis

Edgar Kaziakhmedov
Department of ECE
Binghamton University
Binghamton, NY, USA
ekaziak1@binghamton.edu

Jessica Fridrich
Department of ECE
Binghamton University
Binghamton, NY, USA
fridrich@binghamton.edu

Patrick Bas
Univ. Lille, CNRS, Centrale Lille
UMR 9189 CRIStAL
Lille, France
patrick.bas@cnrs.fr

ABSTRACT

Understanding the mechanisms that lead to false alarms (erroneously detecting cover images as containing secrets) in steganalysis is a topic of utmost importance for practical applications. In this paper, we present evidence that a relatively small number of pixel outliers introduced by the image acquisition process can skew the soft output of a data driven detector to produce a strong false alarm. To verify this hypothesis, for a cover image we estimate a statistical model of the acquisition noise in the developed domain and identify pixels that contribute the most to the associated likelihood ratio test (LRT) for steganography. We call such *cover* elements LIEs (Locally Influential Elements). The effect of LIEs on the output of a data-driven detector is demonstrated by turning a strong false alarm into a correctly classified cover by introducing a relatively small number of "de-embedding" changes at LIEs. Similarly, we show that it is possible to introduce a small number of LIEs into a strong cover to make a data driven detector classify it as stego. Our findings are supported by experiments on two datasets with three steganographic algorithms and four types of data driven detectors.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Image manipulation**; **Neural networks**;

KEYWORDS

Steganalysis, false alarm, LIE, acquisition noise, CNNs

ACM Reference Format:
Edgar Kaziakhmedov, Jessica Fridrich, and Patrick Bas. 2025. Effect of Acquisition Noise Outliers on Steganalysis. In *Proceedings of the 2025 ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '25), June 18–20, 2025, San Jose, California.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.XXXX/XXXXXXX.XXXXXXX

1 INTRODUCTION

In steganalysis, it is important to control the false alarm rate for a number of reasons. First, high false alarm rates make it increasingly more difficult for the steganalyst to identify the steganographer in, e. g., a social network due to the large number of users and images that need to be analyzed and the immense diversity of the cover source. Second, images identified as containing steganographic content are likely to be further inspected to determine the steganographic method or software used for hiding the secret and possibly search for the stego key in the ultimate quest to recover the secret message itself. Since such forensic analysis is expensive, high false alarm rates strain computational resources and make practical steganalysis less operative.

Modern steganography detectors are data-driven and can be trained to address a wide spectrum of steganalysis tasks, including the detection of a specific steganographic technique (binary classification), classification of images into multiple classes based on the embedding method (universal blind detectors) [14, 15, 17], estimation of the size of the secret payload (quantitative detectors) [11, 16], and feature extraction for clustering to identify the steganographer from among many actors [12]. These detectors can be roughly divided into two groups – systems trained on low-dimensional representations of images (features) called rich media models [4, 8, 13] and end-to-end trained systems implemented with deep convolutional neural networks (CNNs) [2, 18, 21–23, 25].

In this paper, we take a closer look at what makes modern data-driven steganalyzers classify a cover image as a strong stego. We present evidence that the randomness of the acquisition process itself (the sensor noise) at pixels with low cost has a major effect on the misclassification. We work with datasets for which one can accurately estimate the statistical model of the acquisition noise in the developed domain. The model is used to identify locally influential cover elements (LIEs) by their outlier values of the associated likelihood ratio test for steganography. De-embedding a relatively small number of such LIEs makes data driven detectors correctly classify the image as cover. Vice versa, making a small number of embedding changes at LIEs can turn a strong cover into a false alarm. We emphasize that LIEs are identified purely from the acquisition model and the embedding costs with no feedback from the trained detector. This provides indirect evidence that data driven detectors must be inherently aware of the acquisition model and that they approximate in some sense the most powerful detector.

The paper is structured as follows. In the next section, we briefly review relevant prior art. In Section 3, we describe two datasets used for all experiments. They were selected judiciously in order to have a tight fitting model of the acquisition noise in the developed domain. This model and a method for estimating its parameters are described in Section 4. Section 5 contains the details of two kinds of experiments, the deletion and insertion tests, that are used in this paper to demonstrate the effect of LIEs on soft output of a data driven steganalyzer. The results of all experiments, including their discussion appear in Section 6. The paper is concluded in Section 7, where we also outline possible future extensions of this work.

IH&MMSec '25, June 18–20, 2025, San Jose, California

ACM ISBN 978-1-XXXX-XXXX-XXXXXXXX.
https://doi.org/10.XXXX/XXXXXXX.XXXXXXX

2 RELEVANT PRIOR ART

The authors of [24] showed that CNNs trained for certain steganography methods in the JPEG domain can identify a JPEG image embedded with a secret from a small number of embedding changes, sometimes even a single change—the stego images contain Locally Detectable Embedding Artifacts or LDEAs. For the same source of cover images and network architecture, some steganographic methods do not introduce LDEAs (J-UNIWARD [10]) while others do (J-MiPOD [5]). LDEAs exist for CNNs because they are better equipped to "see" *local* embedding artifacts [24] than steganalyzers based on rich models. This is because CNNs detect steganographic changes via arrays of repeated convolutions transformed by non-linear activations and thus have the ability to reserve some feature maps to communicate outlier values (local artifacts). In contrast, for classifiers that work with rich models formed by *global* higher-order statistics of image noise residuals, the effects of a few localized embedding artifacts will get "drowned" in such global descriptors.

The current paper can be thought of as a complement of the work on LDEAs. We study what triggers a detector to respond with "stego" on *cover* images in which case the detector outputs a false alarm. Are these false alarms perhaps due to a small number of pixels in the cover image similar to LDEAs? Just like for LDEAs, the effect of LIEs on a detector output will generally depend on the cover source, the detector, and the embedding algorithm. LIEs, however, are fundamentally very different from LDEAs. LDEAs are introduced by the embedding algorithm while LIEs naturally occur in cover images. While LIEs could be introduced by the content (the scene itself), we will mostly be interested in LIEs due to the acquisition process, the process of taking a picture. We limit ourselves to the spatial domain in this paper because acquisition noise has a stronger effect in this domain than in the JPEG domain due to quantization. Having said this, LIEs might be present for high JPEG qualities. We leave this part of the study for future research.

3 DATASETS WITH A KNOWN ACQUISITION MODEL

In order to study how pixel outliers due to acquisition contribute to false alarms, we need a dataset with a known acquisition noise model in the developed domain. Undoubtedly, the most realistic dataset would be formed by actually taking multiple images of the same scene from a tripod-mounted camera with fixed settings. However, even with a sturdy tripod one cannot completely avoid small spatial misalignments due to vibrations and variations in e. g. the exposure time [7]. To eliminate the effect of these inevitable imperfections, we decided to form the dataset from single exposures and instead *simulate* multiple exposures of the same scene taken at a higher ISO by adding different realizations of the heteroscedastic sensor noise in the RAW domain. Our "multiple acquisitions" of the same scene thus appear to be taken at a higher ISO. As shown by Bas in his Natural Steganography [1, 20], with accurate estimation of the parameters of the heteroscedastic noise, these images simulated to be taken at a higher ISO are practically statistically indistinguishable from images actually acquired at the higher ISO setting.

In particular, we took $N_{scene} = 503$ RAW (NEF) ISO 64 images (scenes) with a full frame 45MP Nikon Z7. All images were taken outdoors and consist mostly of landscape scenes. By taking additional test pictures of a uniform background, we estimated the heteroscedastic noise model in the RAW domain for ISO 64 and 200. The heteroscedastic model assumes that pixels in the RAW domain are realizations of independent Gaussian random variables $\mathcal{N}(\mu_k^{\text{RAW}}, v_k^{\text{RAW}})$, where μ_k^{RAW} is the noise-free pixel at location k and the variance

$$v_k^{\text{RAW}} = a_{\text{ISO}}\mu_k^{\text{RAW}} + b_{\text{ISO}}, \tag{1}$$

where the two parameters $a_{\text{ISO}}, b_{\text{ISO}}$ depend on the ISO setting. Pronouncing the ISO 64 image as the noise-free image μ_k^{RAW}, simulating acquisitions at ISO > 64 amounts to replacing μ_k^{RAW} with a sample from

$$\mathcal{N}\left(\mu_k^{\text{RAW}}, (a_{\text{ISO}} - a_{64})\mu_k^{\text{RAW}} + b_{\text{ISO}} - b_{64}\right). \tag{2}$$

Both datasets described below were created for ISO 200 with $a_{200} = 0.8298$ and $b_{200} = -821.862$ ($a_{64} = 0.299$ and $b_{64} = -311.112$).

The fact that we simulate multiple RAW exposures allows us to generate as many of them as needed for an accurate estimation of the model in the developed domain. The RAW images were subsequently developed using two different development pipelines (explained below) and cut into $N_{tile} = 10 \times 16 = 160$ non-overlapping 256×256 tiles of floats (to better estimate the model in the developed domain) and finally quantized to 8-bit grayscale to obtain the actual cover images. Since there are N_{tile} tiles per each 45MP scene, we would have in total $N_{scene} \times N_{tile} = 80,480$ cover images. However, we reject cover images with poor content (e. g., tiles with just sky) based on the following content-complexity measure. Each image is transformed as in JPEG compression using block-DCT on disjoint 8×8 blocks. The content complexity measure is the sum of squares of all DCT coefficients from the upper left quadrant of each block (DCT modes (u, v), $0 \le u, v \le 3$, $(u, v) \neq (0, 0)$). Then all images are ranked w.r.t. their complexity score and we discarded $23,000$ images[1] with the lowest score. Thus, the actual number of grayscale 256×256 cover images in our datasets was $N_{set} = 57,480$.

The two development pipelines are:

(1) **SENSED**. The development of this dataset does not include demosaicking. Each 2×2 Bayer filter square with one red, one blue, and two green RAW values has been converted to a single colored RAW pixel to simulate actually sensing all three colors at every pixel.[2] This subsampling step changes the 45MP RAW file to a new 11.25MP RAW file in which all three colors are registered at each pixel. This RAW file is further processed with Python's `rawpy` library (raw colorspace, 1008 black level, all automatic adjustments turned off) to obtain a 16-bit color image, which is then converted to grayscale floats in the interval $[0, 255]$ (for developed-domain acquisition model estimation) and quantized to 8 bits. The resulting image is cut into N_{tile} non-overlapping 256×256 8-bit grayscale images forming the dataset SENSED.

(2) **VNG**. This dataset is more realistic. The 45MP RAW file was processed using VNG color interpolation (raw colorspace, 1008 black level, with all automatic adjustments disabled) to

[1] This value was selected based on visual inspection and was fixed for both datasets described below in order to have the same number of images in both datasets.

[2] The green pixel was selected from the same position in the 2×2 Bayer tile across all 2×2 tiles (lower left pixel).

produce a 16-bit color image. This image was then converted to grayscale floats within the range $[0, 255]$ for estimating the developed-domain acquisition model, before being quantized to 8-bit grayscale. The resulting image is downsampled by a factor of 2 using bicubic interpolation and divided into the same number of non-overlapping 256×256 tiles.

We note that the rejection of images based on content complexity described above has been executed for the VNG dataset and was then adopted for the SENSED dataset as well. The first dataset SENSED was included for its simplicity. Since there is no demosaicking, this development pipeline does not introduce dependencies among pixels in the developed domain. Due to the Gaussianity of the heteroscedastic model, the developed domain model is a multivariate Gaussian (MVG) with a diagonal covariance. As explained and detailed in the next section, we will use this model in the developed domain for the more realistic VNG dataset as well. Since the VNG color interpolation algorithm and the subsequent downscaling introduce dependencies among neighboring pixels, the MVG model with a diagonal covariance is only an approximation for this dataset.

4 DEVELOPED DOMAIN MODEL

We adopt a particularly simple Multivariate Gaussian (MVG) model with a diagonal covariance for the pixels in the developed domain. While this model is only an approximation for VNG due to the dependencies introduced by color interpolation and downscaling, this model ideally lends itself for identifying individual outlier cover elements (LIEs). If we were to work with an entire Markov random field, the concept of a LIE and its identification would have to be properly generalized to small pixel neighborhoods. We leave this option for future research.

Having generated $B+1$ simulated exposures in the developed domain, the mean and variance are computed from the unquantized cover pixels (their float versions) to decrease the estimation error. Formally, we start with $B+1$ images (acquisitions of the ith scene) $\mathbf{x}^{(i,j)} \in \mathbb{R}^{256\times 256}$, where $i \in \{1, \dots, N_{scene}\}$ is the cover scene index and $j \in \{1, \dots, B+1\}$ is the index of the simulated burst. Selecting the first image in the "burst" as the cover $\mathbf{x}^{(i)} = \mathbf{x}^{(i,1)}$, we use the remaining B images $\mathbf{x}^{(i,2)}, \dots, \mathbf{x}^{(i,B+1)}$ to estimate the mean and variance of the (k,l)th pixel, $1 \le k, l \le 256$:

$$\mu_{kl}^{(i)} = \frac{1}{B}\sum_{j=2}^{B+1} \mathbf{x}_{kl}^{(i,j)} \tag{3}$$

$$v_{kl}^{(i)} = \frac{1}{B-1}\sum_{j=2}^{B+1} \left(\mathbf{x}_{kl}^{(i,j)} - \mu_{kl}^{(i)}\right)^2. \tag{4}$$

Since the pixel values are 8-bit integers from $\mathcal{I}_8 = \{0, 1, \dots, 255\}$, the model of the k,lth cover pixel in the 8-bit grayscale developed domain is the Gaussian $\mathcal{N}(\mu_{kl}^{(i)}, v_{kl}^{(i)})$ floored to integers and clipped to a finite dynamic range at 0 and 255

$$x_{kl}^{(i)} \sim p_{kl}^{(i)} \triangleq \left\lfloor \mathcal{N}(\mu_{kl}^{(i)}, v_{kl}^{(i)}) \right\rfloor, \tag{5}$$

where $\lfloor x \rfloor$ is the operation of flooring $x \in \mathbb{R}$ and

$$p_{kl}^{(i)}(m) = \begin{cases} 1 - Q_{kl}^{(i)}(m+1) & m = 0 \\ Q_{kl}^{(i)}(m) - Q_{kl}^{(i)}(m+1) & 1 \le m \le 254 \\ Q_{kl}^{(i)}(m) & m = 255 \end{cases} \tag{6}$$

with $Q_{kl}^{(i)}(x)$ defined as the tail probability of $\mathcal{N}(\mu_{kl}^{(i)}, v_{kl}^{(i)})$:

$$Q_{kl}^{(i)}(x) \triangleq \mathbb{P}\{\mathcal{N}(\mu_{kl}^{(i)}, v_{kl}^{(i)}) > x\}. \tag{7}$$

5 LIES

In this paper, we will assume that the Warden has an empirical detector, for example a deep CNN, trained to detect embedding with a known steganographic algorithm. With deep CNNs, we take the stego class soft output as the detector's output. Formally, CNNs output two unnormalized class scores called the cover and stego logits, z_0 and z_1, respectively. We take the stego logit class as the detector's output z, $z : \mathcal{I}_8^{256\times 256} \to \mathbb{R}$. After applying the `softmax` function, the mapping becomes $d : \mathcal{I}_8^{256\times 256} \to [0,1]$, $d = e^{-z_1}/(e^{-z_0} + e^{-z_1})$, representing a normalized soft output for stego class (normalized soft output for cover class is omitted). The Warden makes binary decisions on images by thresholding d with threshold t,

$$d(\mathbf{y}) > t \implies \mathbf{y} \text{ is a stego image.} \tag{8}$$

The threshold is typically set in practice to achieve a desired false alarm

$$\mathbb{P}\{d(\mathbf{x}) > t | \mathbf{x} \text{ cover}\} = P_{\text{FA}} \tag{9}$$

or set to $t = 0.5$ for the Bayesian detector that minimizes the total error probability under equal priors

$$P_{\text{E}} = \frac{1}{2}(P_{\text{D}} + P_{\text{FA}}),$$

where P_{D} is the detector power (true positive rate).

As explained in the introduction, we hypothesize that strong false alarms are caused by a relatively small number of outlier pixels or LIEs that skew the output of Warden's detector. If the detector misinterprets LIEs as embedding, then we need to take embedding into account when looking for LIEs. In particular, it is not enough for a pixel to be an outlier w.r.t. the acquisition noise in the developed domain but it also needs to be in an area where pixels are likely to be modified by embedding. Our algorithm for finding LIEs only uses the acquisition model and the embedding change rates. In particular, we make no use of the trained detector.[3]

We start with Warden's hypothesis test formulated within the adopted acquisition model, derive the most powerful steganography detector within the model (the likelihood ratio test), and identify LIEs based on the largest values of the LRT. Formally, let $\rho_{kl}^{(i)} \ge 0$ denote the symmetric embedding costs computed from the ith cover image. The costs are computed by the steganographic method used for embedding. The embedding modifications $s_{kl} \in \{-1, 0, 1\}$, form an array of independent samples from a ternary random variable attaining values in $\{-1, 0, +1\}$ with probabilities $\beta_{kl}^{(i)}, 1 - 2\beta_{kl}^{(i)}, \beta_{kl}^{(i)}$,

[3] This is in contrast with [11] where the authors used attribution maps of d (vanilla or integrated gradients) to find influential pixels in covers. While influential pixels found this way affect the output more, they may contain an adversarial component as they are found with feedback from the detector.

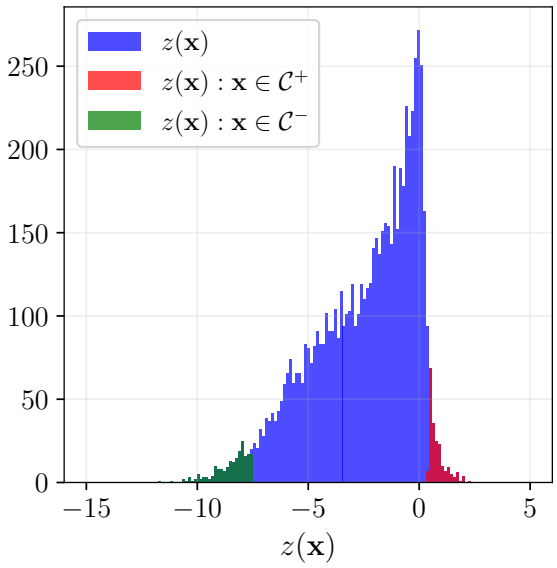

Figure 1: Distribution of stego logits $z(\mathbf{x})$ for cover images $\mathbf{x}$. Stego logit $z(\mathbf{x})$ is used instead of soft output $d(\mathbf{x})$ for demonstration purposes, as softmax compresses the differences and reduces visualization clarity. Red represents the logit values of images from $\mathcal{C}^+$, while green corresponds to the logits of images from $\mathcal{C}^-$. S-UNIWARD, SENSED dataset, SRNet trained for a fixed payload of 0.4 bpp.

respectively, which are determined by the payload and the embedding costs of the ith cover. The stego pixel probability mass function (p.m.f.) $q_{kl}^{(i)}$ is a mixture of quantized Gaussians for all pixels kl :

$$\begin{aligned} q_{kl}^{(i)}(m) = (1 - 2\beta_{kl}^{(i)})p_{kl}^{(i)}(m) + \\ + \beta_{kl}^{(i)} p_{kl}^{(i)}(m-1) \\ + \beta_{kl}^{(i)} p_{kl}^{(i)}(m+1), \end{aligned} \tag{10}$$

with proper truncation at the boundaries of the dynamic range $m \in \{0, 255\}$.

Given one specific image $\mathbf{y}^{(i)} \in \mathcal{I}_8^{256\times256}$, the steganalyst needs to decide whether its pixels follow the cover or stego distributions through the following statistical hypothesis test for all kl :

$$\begin{aligned} \mathcal{H}_0 &: y_{kl}^{(i)} \sim p_{kl}^{(i)} \\ \mathcal{H}_1 &: y_{kl}^{(i)} \sim q_{kl}^{(i)}. \end{aligned} \tag{11}$$

For this test, we will assume that the model parameters, the means $\mu_{kl}^{(i)}$ and the variances $v_{kl}^{(i)}$, as well as the change rates $\beta_{kl}^{(i)}$ are known. Hence the test is simple, and, by the statistical independence of pixels in both cover and stego images, the most powerful detector is the log-likelihood ratio

$$\Lambda^{(i)}(\mathbf{y}^{(i)}) = \sum_{kl} \Lambda_{kl}^{(i)}(y_{kl}^{(i)}) = \sum_{kl} \log\left(\frac{q_{kl}^{(i)}(y_{kl}^{(i)})}{p_{kl}^{(i)}(y_{kl}^{(i)})}\right), \tag{12}$$

where $\Lambda_{kl}^{(i)}(m) = \log(q_{kl}^{(i)}(m)/p_{kl}^{(i)}(m))$, $m \in \mathcal{I}_8$.

Excluding pixels with wet costs or with variance $v_{kl}^{(i)} \leq 0.01$, LIEs correspond to pixels with the largest values of $\Lambda_{kl}^{(i)}(y_{kl}^{(i)})$, $1 \leq k, l \leq 256$. This means that LIEs are acquisition noise outliers in places where the embedding is likely to make an embedding change (low cost). Based on this criterion, we next describe two kinds of experiments (tests) that assess the effect of LIEs on the soft output of a data driven steganalyzer.

5.1 Deletion test

In the deletion test, which we sometimes call "de-embedding test," we suppress LIEs one by one to make a cover image identified as stego look progressively more like a cover. Given a cover image $\mathbf{x}^{(i)}$ detected as a strong false alarm, its pixels are sorted by $\Lambda_{kl}^{(i)}(x_{kl}^{(i)})$. Then, these pixels are modified

$$x_{kl}^{(i)} \to x_{kl}^{(i)} + \xi_{kl}^{(i)}, \text{ where} \tag{13}$$

$$\xi_{kl}^{(i)} = \arg\min_{\xi\in\{-1,1\}} \left\{\Lambda_{kl}^{(i)}(x_{kl}^{(i)} + \xi)\right\}. \tag{14}$$

The idea is to modify the pixels by ±1 (de-embed) in such a way that decreases the Warden's detection statistic to make them look more like covers, and observe how the output of Warden's empirical (trained) detector changes with more pixels being de-embedded. We remind that just because the de-embedding changes are selected to decrease the LRT of the cover does not automatically mean that the output of a trained empirical detector will consistently decrease with each de-embedding change because the empirical detector is unlikely to coincide with the theoretically most powerful detector. If it does so most of the times, it constitutes some evidence that the empirical detector is aware of the acquisition model.

5.2 Insertion test

This is a complement of the deletion test. Here, we begin with a cover image strongly identified as cover (taken from the left tail of the detector's output distribution) and introduce embedding changes at pixels so that the LRT increases the most. In other words, we intentionally introduce LIEs. Formally, we first compute for each i, k, l

$$\eta_{kl}^{(i)} = \arg\max_{\eta\in\{-1,1\}} \left\{\Lambda_{kl}^{(i)}(x_{kl}^{(i)} + \eta)\right\} \tag{15}$$

and then order the pixels by $\Lambda_{kl}^{(i)}(x_{kl}^{(i)} + \eta_{kl}^{(i)})$ from the largest to the smallest and replace one-by-one $x_{kl}^{(i)} \to x_{kl}^{(i)} + \eta_{kl}^{(i)}$. Again, we observe how these "embedding changes" affect the soft output of Warden's empirical detector.

6 EXPERIMENTS

The existence, strength, and role of LIEs will likely depend on the dataset, the detector, and the embedding algorithm. To obtain a more comprehensive picture, in this section we work with two datasets described above, three types of detectors (SRNet [2] trained for a fixed payload, for a uniform mixture of payloads, and a quantitative detector), and for three embedding algorithms, S-UNIWARD [10], MiPOD [19], and non-adaptive Least Significant Bit Matching (LSBM). Finally, we also study whether LIEs affect the older generation of detectors implemented as classifiers with rich media models.

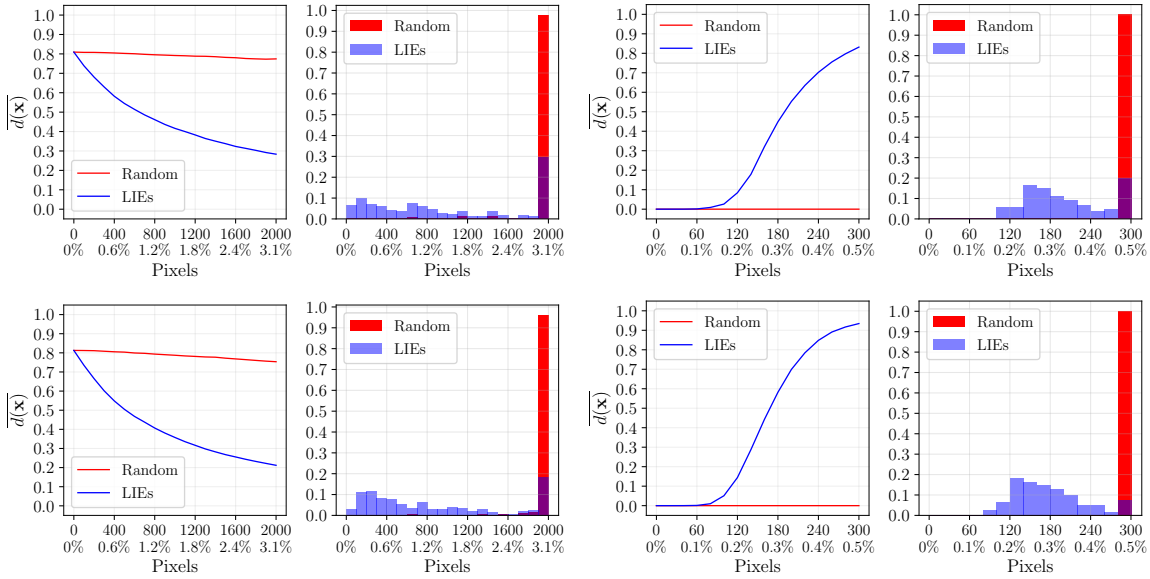

Figure 2: The effect of modifying cover pixels in strong false alarms C^+ (left pair of subplots) and in strong covers C^- (right pair) as described in the deletion and insertion tests, respectively. The left charts show the output $\overline{d(\mathbf{x})}$ averaged over the corresponding subsets of covers x. The normalized histograms display the number of pixels that had to be modified to bring the output below 0.5 (for false alarms) and above 0.5 (for strong covers). Blue corresponds to the actual deletion and insertion tests while red is used for the same experiment but with a random selection of pixels instead. Top: SENSED, Bottom: VNG. S-UNIWARD, SRNet trained for a fixed payload of 0.4 bpp.

The deletion and insertion tests were run for the 3% right and 3% left detector's output outliers – cover images identified strongly as stego (false alarms) and images identified strongly as covers. For both SENSED and VNG datasets 3% corresponds to 206 images. Formally, cover images $\mathbf{x}$ selected for the deletion test came from the subset

$$C^+ = \{\mathbf{x} | d(\mathbf{x}) > t^+\} \tag{16}$$

with the threshold t^+ determined from the condition

$$\mathbb{P}\{d(\mathbf{x}) > t^+ | \mathbf{x} \text{ cover}\} = 0.03, \tag{17}$$

and for the insertion test from

$$C^- = \{\mathbf{x} | d(\mathbf{x}) < t^-\} \tag{18}$$

$$\mathbb{P}\{d(\mathbf{x}) < t^- | \mathbf{x} \text{ cover}\} = 0.03. \tag{19}$$

Then, the pixels in these images were modified one by one as described for the respective tests in the previous section. After each modification, they were run through the detector d to observe the effect on the output. Figure 1 is a visual interpretation of how the sets C^-, C^+ are formed based on the detector's output distribution.

Figure 2 shows the results of the deletion and insertion tests on SENSED (top) and VNG (bottom) datasets. The left subplots show the output $d(\mathbf{x})$ averaged over covers from C^+ (top) and C^- (bottom) as a function of the modified pixels (blue). The right subplots show the normalized histogram of the number of pixels needed to change the decision of the Bayesian detector (to bring the output below 0.5 for deletion and above 0.5 for insertion). Red color in these plots is used for the same test (the pixels are modified in the same fashion) but when the pixels are selected at random from the images. This serves as a baseline. The detector is the SRNet [2] trained for S-UNIWARD with stego images embedded with a fixed payload 0.4 bpp. The details of the network training and the training, validation, and test sets are described in Section 6.2.

The figure clearly shows that selecting the pixels based on the LRT rather than randomly is much more effective. For example, on average for the SENSED dataset, 1.03% of pixels (674 pixels) need to be de-embedded to bring the output below 0.5. On the other hand, when introducing LIEs as in the insertion test, fewer than 0.3% of pixels (190 pixels) need to be modified in covers strongly identified as cover (from C^-) to make them into a false alarm for the Bayesian detector. In contrast, changing the same number of randomly selected pixels (red curve) has virtually no effect on the detector's output.[4] For randomly selected pixels, we observed that on average 24.97% (15,707) pixels must be de-embedded to decrease the output below 0.5 in the deletion test, whereas 3.06% of the pixels (2,003 pixels) need to be modified to increase the output above 0.5 in the insertion test. For an informative histogram of LIEs, we clipped the histograms for randomly selected pixels. These results confirm our intuition that acquisition noise outliers as identified by Warden's LRT are indeed influential (LIEs).

Despite the fact that the MVG model with a diagonal covariance is only an approximation for the VNG dataset (in contrast to the SENSED dataset), the LIEs seem to have a stronger impact on the detector's output in terms of the number of pixels needed to achieve the respective goals in the deletion and insertion tests.

Figure 3 shows the actual output $d(\mathbf{x})$ for each image from $\mathbf{x} \in C^+$ ($\mathbf{x} \in C^-$) as a function of the number of modified pixels in the

[4]We note that an image embedded with payload 0.4 bpp on average contains 6% of modified pixels (4000).

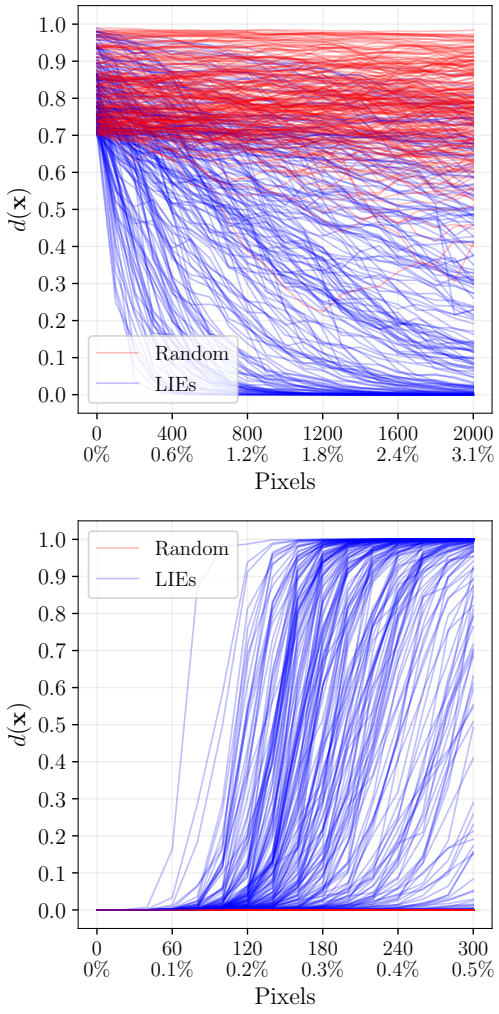

Figure 3: The actual rather than averaged outputs $d(\mathbf{x})$ across images $\mathbf{x}$ from C^+ (top) and C^- (bottom) for the deletion and insertion tests reported in Figure 2.

deletion (top) and insertion (bottom) tests (instead of the averaged outputs) for the same setup as in Figure 2. This is informative because it shows that not all changes lead to a decrease of the detector's output. This is because the empirical detector does not coincide with the LRT used to identify LIEs. In general, however, suppressing / introducing LIEs does have the expected effect.

In Figure 4, we show an example of a textured image (top) and a smooth image (bottom) from the VNG dataset with LIEs to be de-embedded highlighted in yellow. The top subplot clearly communicates that LIEs are more likely to be in the textured areas of the image, i. e., with sufficiently large β_{kl}. Conversely, in the bottom subplot we observe that LIEs are more likely to cluster around pixels with a small noise variance v_{kl}. This is because in this case the quantization to 8 bits has a larger effect on an unquantized acquisition noise outlier—it can in fact suppress it or make it an ever larger outlier. In the latter case, when $v_{kl} < 1$ the quantized value may thus look like an embedding change.

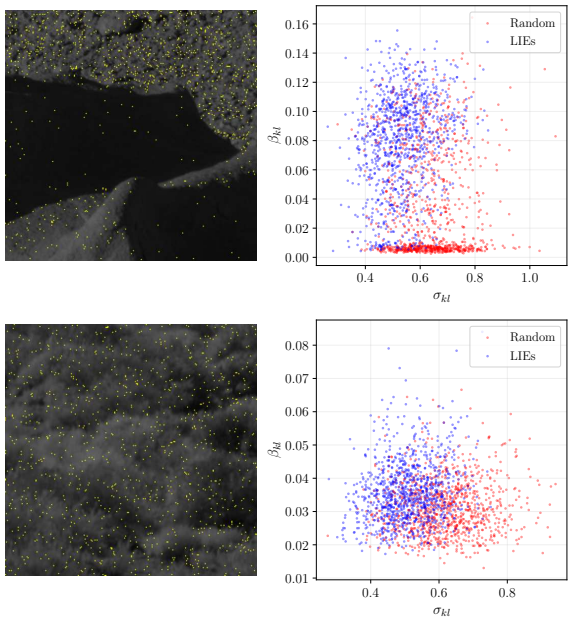

Figure 4: The distribution of 1000 LIEs with the largest LRT highlighted in yellow (deletion test) in two cover images from the VNG dataset. LIEs are pixel outliers w.r.t. the acquisition noise in areas with low embedding costs and small sigma (see the scatter plots). The LRT was evaluated for S-UNIWARD. The two cover images were selected from C^+ of SRNet trained on a fixed payload 0.4 bpp.

In the following set of experiments, we studied how LIEs are affected by the embedding algorithm. To this end, we executed the same experiment as the one reported in Figure 2 but with the VNG dataset, SRNet trained for a fixed payload 0.4, and non-adaptive LSBM and MIPOD. As shown in Figure 5, contrasting the figures for LSBM (top) and the content adaptive MiPOD (bottom) and S-UNIWARD (Figure 2, bottom), the effect of LIEs in the deletion test is the strongest for MiPOD, while for the insertion test LIEs are the strongest for S-UNIWARD.

In our final experiment with CNNs, we studied how the deletion and insertion tests are affected by how the detector is trained. Instead of working with a detector trained for a fixed payload, we use a detector trained on a uniform mixture of payloads from the set

$$\mathcal{P} = 0.05, 0.1, 0.15, 0.2, 0.3, \ldots, 1.0 \tag{20}$$

and a quantitative detector that outputs an estimate of the secret payload size. The output of the quantitative detector is denoted $\hat{\alpha} : \mathcal{I}_8^{256\times256} \to \mathbb{R}$ as it represents the unnormalized predicted payload and does not undergo the `softmax` normalization. The steganalysis hypothesis test for such detectors is no longer simple due to the unknown payload. Thus, we need to change the criterion for finding LIEs. To this end, we convert the composite test to a simple test by imposing a prior on the unknown payload – the uniform prior. This prior makes sense since the data driven detectors were trained on a uniform mixture of payloads from $\mathcal{P}$. Mathematically, we replace

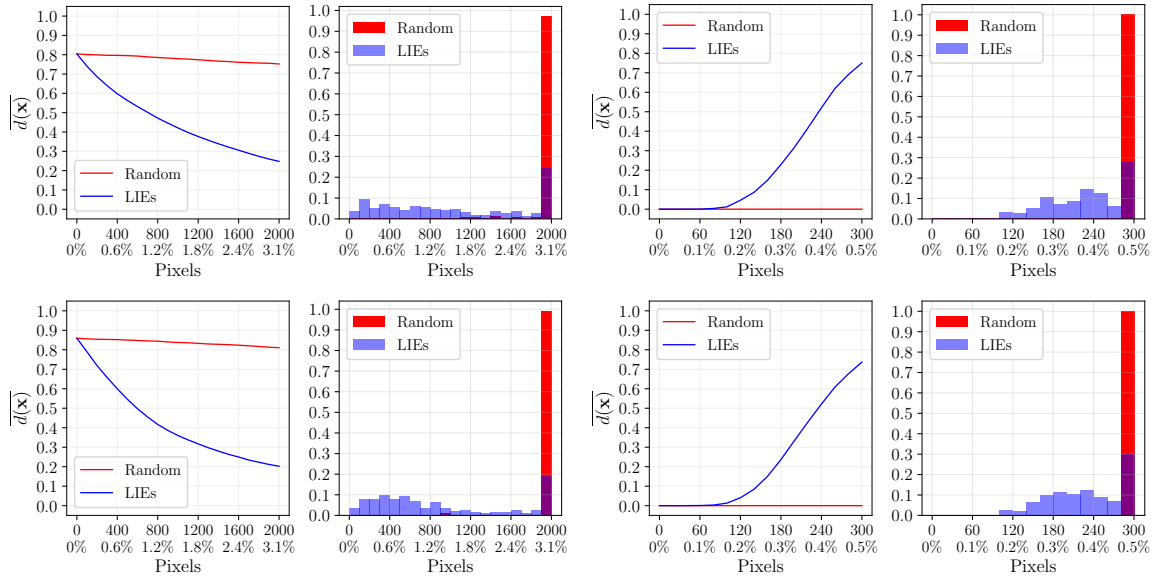

Figure 5: [LSBM and MiPOD] The effect of modifying cover pixels in strong false alarms C^+ (left) and in strong covers C^- (right) as described in the deletion and insertion tests, respectively. The left subplots show the output $\overline{d(\mathbf{x})}$ averaged over the corresponding subsets of covers x. The normalized histograms shows the number of pixels that had to be modified to bring the output below 0.5 (for false alarms) and above 0.5 (for strong covers). Blue corresponds to the actual deletion and insertion tests while red is used for the same experiment but with random selection of pixels instead. Top: LSBM, Bottom: MiPOD, VNG dataset, SRNet trained for a fixed payload of 0.4 bpp.

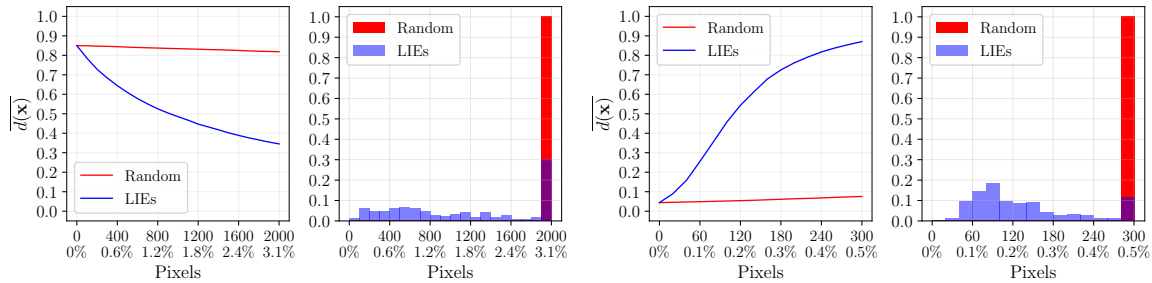

Figure 6: [Detector trained on uniform payloads] The effect of modifying cover pixels in strong false alarms C^+ (left) and in strong covers C^- (right) as described in the deletion and insertion tests, respectively. The left charts show the output $\overline{d(\mathbf{x})}$ averaged over the corresponding subsets of covers x. The normalized histograms show the number of pixels that needed to be modified to bring the output below 0.5 (for false alarms) and above 0.5 (for strong covers). Blue corresponds to the actual deletion and insertion tests while red is used for the same experiment but with random selection of pixels instead. S-UNIWARD, VNG dataset, SRNet trained on a uniform mixture of payloads from $\mathcal{P}$.

the LRT $\Lambda_{kl}^{(i)}(m) = q_{kl}^{(i)}(m)/p_{kl}^{(i)}(m)$ with

$$\Lambda_{kl}^{(i)}(m) = \overline{q_{kl}^{(i)}}(m)/p_{kl}^{(i)}(m), \tag{21}$$

where $\overline{q_{kl}^{(i)}}(m)$ is the stego mixture averaged over all payloads from $\mathcal{P}$.

Figure 6 displays the results for S-UNIWARD, the VNG dataset, and SRNet trained on a uniform mixture of payloads from $\mathcal{P}$. We observe that the deletion test for this detector is slightly less successful than for a detector trained on a fixed payload (Figure 2 bottom left). We attribute it to the fact that the detector trained on a range of payloads is more sensitive to embedding since it sees smaller payloads during training. Thus de-embedding does not have such a strong effect. On the other hand, the insertion test works better since fewer changes are enough to trigger such a detector (the blue curve is steeper than in Figure 2 bottom right).

Figure 7 displays the results for S-UNIWARD, VNG dataset, and a quantitative SRNet detector implemented as in [11]. Unlike binary detectors, a quantitative detector is trained to increase its output proportionally w.r.t. embedded payload size. Additionally, its output is expected to be contained within a certain range. Thus, the deletion and insertion tests are significantly less sensitive to input

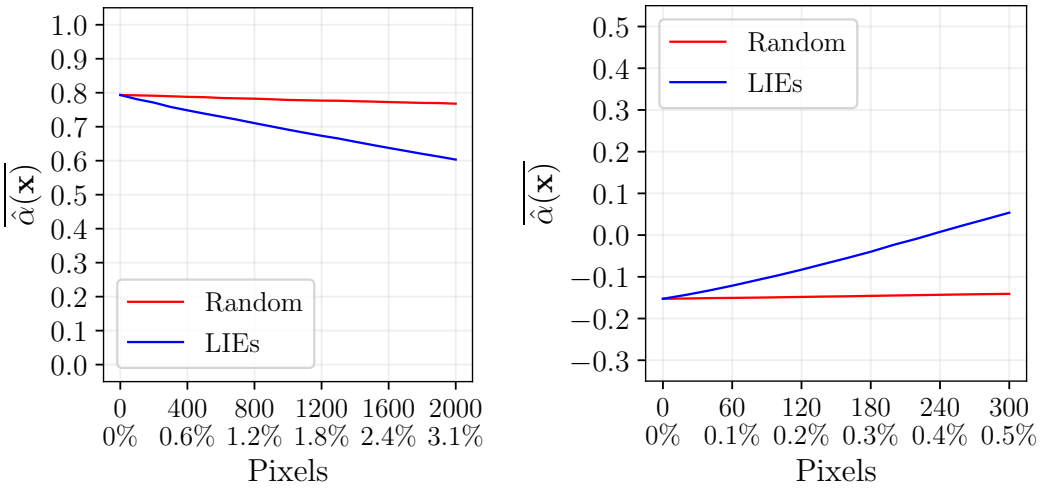

Figure 7: [Quantitative detector] The effect of modifying cover pixels in strong false alarms C^+ (left) and in strong covers C^- (right) as described in the deletion and insertion tests, respectively. The charts show the output $\overline{\hat{\alpha}(\mathbf{x})}$ averaged over the corresponding subsets of covers $\mathbf{x}$. Blue corresponds to the actual deletion and insertion tests while red is used for the same experiment but with random selection of pixels instead. S-UNIWARD, VNG dataset, SRNet trained as a quantitative detector.

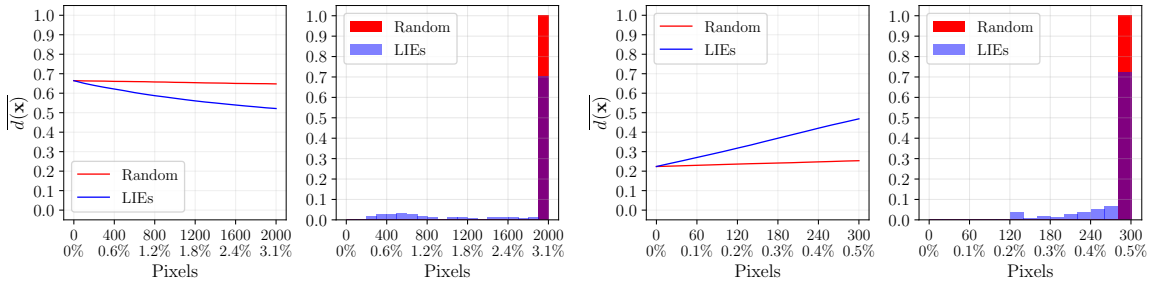

Figure 8: [SRMQ1 with LCLC] The effect of modifying cover pixels in strong false alarms C^+ (left) and in strong covers C^- (right) as described in the deletion and insertion tests, respectively. The left subplots show the output $\overline{d(\mathbf{x})}$ averaged over the corresponding subsets of covers $\mathbf{x}$. The right subplots are normalized histograms of the number of pixels that needed to be modified to bring the output below 0.5 (for false alarms) and above 0.5 (for strong covers). Blue corresponds to the actual deletion and insertion tests while red is used for the same experiment but with random selection of pixels instead. S-UNIWARD, VNG dataset, SRM with LCLC trained for a fixed payload of 0.4 bpp.

perturbations and the average detector's output is approximately linear w.r.t. the number of modified pixels.

6.1 LIEs for classifiers with rich models

Finally, we investigate whether older detectors implemented as classifiers with rich media models are also affected by LIEs. Our expectation is that they will be affected to a much lesser degree because, unlike CNNs, such detectors are not as well equipped to see local artifacts. Our experiments confirm this expectation. Figure 8 shows the equivalent of Figure 2 (bottom) but executed with the spatial rich model (SRMQ1) [8] and the LCLC classifier [6]. For cover $\mathbf{x}$, we applied the sigmoid function to the projection of its feature vector $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^p$ onto the weight vector $\mathbf{w} \in \mathbb{R}^p$ ($p = 12,753$ for SRMQ1) to give it the same scaling as a network output $d(\mathbf{x})$:

$$d(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{f}(\mathbf{x})\cdot\mathbf{w}}}. \tag{22}$$

While the LIEs do have a larger effect on the detector output than randomly selected pixels (red curves), their effect is much weaker than for the SRNet. In particular, note that for the vast majority of images from C^+ and C^- even the maximal number of changes (2000 and 300) failed to change the image class.

6.2 Detector training

In this subsection, we describe the training of all detectors. Both SENSED and VNG datasets were split into TRN, VAL, and TST subsets with size 80%, 8%, and 12% of total dataset size. For all setups, SRNet was seeded with JIN-SRNet [3] weights. For the fixed-payload detector, SRNet was trained for 100 epochs with the `adamw`

Detector type		SUNI, SENSED	SUNI, VNG	MiPOD, VNG	LSBM, VNG
0.4 bpp SRNet	P_E	0.1641	0.2015	0.1913	0.1728
multi-payload SRNet	P_E	-	0.2525	-	-
LCLC SRMQ1	P_E	-	0.3163	-	-
quantitative SRNet	MSE	-	0.0533	-	-

Table 1: Detection performance on the test set for all detectors used in the experiments. A dash indicates that a detector was not used. Each column corresponds to a specific embedding algorithm and dataset type, while the second column specifies the performance measure. The first three rows are for a binary detector hence the performance is measured with P_E, while the last row corresponds to a quantitative detector evaluated using MSE.

optimizer and learning rate 10^{-3}. The learning rate policy was set to `onecycle`.

For training the detector on a range of payloads on the VNG dataset, we sampled stego images with payloads uniformly randomly from $\mathcal{P}$ while making sure the payloads do not exceed the maximum embedding capacity. During training, we sampled only 10% of the entire training set every epoch. The training parameters are the same as above (for the fixed-payload detector).

In order to train the quantitative detector, we sampled six payloads ranging from 0.001 to 1.5 bpp spaced with 0.001 bpp per each image. The largest payload was again limited by the embedding capacity of each cover image. For training, we also subsample only 10% of the training set every epoch with the same training parameters as above.

Table 1 shows the detection performance of all detectors used in our study.

7 CONCLUSIONS

In this paper, we study the effect of acquisition noise on false alarms of data-driven detectors built as convolutional neural networks and classifiers with rich media models. We present evidence that, for detectors implemented as CNNs, a relatively small number of pixel outliers introduced by the image acquisition process can skew the detector's output to produce a strong false alarm. To verify this hypothesis, we work with a dataset of images for which we can realistically simulate multiple acquisitions of the same scene by adding heteroscedastic (ISO) noise in the RAW domain. This allows us to estimate a statistical model in the developed domain and identify pixels that contribute the most to the likelihood ratio test (LRT) for steganography. We call such cover elements LIEs, Locally Influential Elements. We stress that LIEs are identified from the model and with no feedback from the data driven detector.

To demonstrate that LIEs have a major effect on the network detector output, we execute two kinds of complementary experiments – deletion and insertion tests. In the deletion test, we work with the right tail outliers of the network output (strong false alarms) and "de-embed" the LIEs in these images to decrease the LRT. In the insertion test, we start with a strong cover – a left tail outlier of the network output – and introduce outliers by modifying the pixels by ±1 to introduce the strongest LIEs. In both tests, merely hundreds of changes are needed to make a strong false alarm be classified as cover and to make a strong cover be misclassified as stego. In contrast, executing the same number of de-embedding and embedding changes to randomly selected pixels has virtually no effect on the detector's output.

We widen our study to include three embedding algorithms, the content-adaptive S-UNIWARD, MiPOD, and non-adaptive LSB matching. We also work with two types of datasets, one where the adopted statistical model of acquisition noise in the developed domain is true and one where it is only an approximation. We also study how the way the detector is trained (binary detector for a fixed payload, for a mixture of payloads, and a quantitative detector) affects LIEs and their effect on detection.

In contrast, LIEs have a much weaker effect on detectors trained as classifiers with the spatial rich model. This is to be expected since such detectors use global image descriptors and thus are less sensitive to local artifacts. Of course, this by no means makes these detectors superior to CNNs as they lag behind CNNs in basically every respect.

Finally, we make some comments on LIEs vs. LDEAs [24] (Locally Detectable Embedding Artifacts) and on LIEs in the JPEG domain. Starting with the latter, unlike LDEAs, which are far more likely to be present in the JPEG domain due to the larger energy of steganographic modifications (due to quantization), LIEs are more likely to be present in the spatial domain because the acquisition noise is largely decimated in JPEG images. Having said this, LIEs might be present for the highest quality factors. Based on our analysis, which was confined to the spatial domain, we have not seen much difference in terms of LIEs presence and their strength w.r.t. the embedding algorithm. The cost-based S-UNIWARD, the model-based MiPOD, and the non-adaptive LSBM all exhibited LIEs and the network detectors trained to detect them were similarly responsive to LIEs.

Our future directions include using more complex models of the acquisition noise in the developed domain (Markov Random Fields) to better identify influential cliques of pixels in datasets with strong dependencies among neighboring pixels. This would require rethinking the concept of a LIE to a group of pixels. Also, LIEs could be used as a means to understand how a data driven detector works. Say, a detector utilizes a compatibility or soft compatibility among pixel noise residuals for detection (as it is the case for e. g. "freshly interpolated" color images [9]). Then, identifying pixels based on a stego mixture for pixels will be less effective than forming the statistic for identifying LIEs through the proper test for the residuals. Once the proper domain and a statistical model is found in terms of

strong LIEs, we will have discovered something fundamental about the inner workings of the trained detector.

ACKNOWLEDGMENTS

The work on this paper was supported by NSF grant No. 2324991.

REFERENCES

[1] P. Bas. Steganography via cover-source switching. In *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, Abu Dhabi, UAE, December 4–7 2016.

[2] M. Boroumand, M. Chen, and J. Fridrich. Deep residual network for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 14(5):1181–1193, May 2019.

[3] J. Butora, Y. Yousfi, and J. Fridrich. How to pretrain for steganalysis. In D. Borghys and P. Bas, editors, *The 9th ACM Workshop on Information Hiding and Multimedia Security*, Brussels, Belgium, June 22–25, 2021. ACM Press.

[4] L. Chen, Y.-Q. Shi, and P. Sutthiwan. Variable multi-dimensional co-occurrence for steganalysis. In *Digital Forensics and Watermarking, 13th International Workshop, IWDW*, volume 9023, pages 559–573, Taipei, Taiwan, October 1–4, 2014. Springer.

[5] R. Cogranne, Q. Giboulot, and P. Bas. Steganography by minimizing statistical detectability: The cases of JPEG and color images. In C. Riess and F. Schirrmacher, editors, *The 8th ACM Workshop on Information Hiding and Multimedia Security*, Denver, CO, 2020. ACM Press.

[6] R. Cogranne, V. Sedighi, T. Pevný, and J. Fridrich. Is ensemble classifier needed for steganalysis in high-dimensional feature spaces? In *IEEE International Workshop on Information Forensics and Security*, Rome, Italy, November 16–19, 2015.

[7] T. Denemark and J. Fridrich. Steganography with multiple JPEG images of the same scene. *IEEE Transactions on Information Forensics and Security*, 12(19):2308–2319, October 2017.

[8] J. Fridrich and J. Kodovský. Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 7(3):868–882, June 2011.

[9] M. Goljan and J. Fridrich. CFA-aware features for steganalysis of color images. In A. Alattar and N. D. Memon, editors, *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2015*, volume 9409, San Francisco, CA, February 8–12, 2015.

[10] V. Holub, J. Fridrich, and T. Denemark. Universal distortion design for steganography in an arbitrary domain. *EURASIP Journal on Information Security, Special Issue on Revised Selected Papers of the 1st ACM IH and MMS Workshop*, 2014:1, 2014.

[11] E. Kaziakhmedov, E. Dworetzky, and J. Fridrich. Analyzing quantitative detectors for content-adaptive steganography. In A. Alattar and N. D. Memon, editors, *Proceedings IS&T, Electronic Imaging, Media Watermarking, Security, and Forensics 2024*, San Francisco, CA, January 21–25, 2024.

[12] A. D. Ker and T. Pevný. The steganographer is the outlier: Realistic large-scale steganalysis. *IEEE Transactions on Information Forensics and Security*, 9(9):1424–1435, September 2014.

[13] J. Kodovský and J. Fridrich. Steganalysis of JPEG images using rich models. In A. Alattar, N. D. Memon, and E. J. Delp, editors, *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2012*, volume 8303, pages 0A 1–13, San Francisco, CA, January 23–26, 2012.

[14] T. Pevný and J. Fridrich. Towards multi-class blind steganalyzer for JPEG images. In M. Barni, I. J. Cox, T. Kalker, and H. J. Kim, editors, *International Workshop on Digital Watermarking*, volume 3710 of Lecture Notes in Computer Science, Siena, Italy, September 15–17, 2005. Springer-Verlag, Berlin.

[15] T. Pevný and J. Fridrich. Multiclass blind steganalysis for JPEG images. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, pages 257–269, San Jose, CA, January 16–19, 2006.

[16] T. Pevný, J. Fridrich, and A. D. Ker. From blind to quantitative steganalysis. *IEEE Transactions on Information Forensics and Security*, 7(2):445–454, 2011.

[17] T. Pevný and A. D. Ker. The challenges of rich features in universal steganalysis. In A. Alattar, N. D. Memon, and C. Heitzenrater, editors, *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2013*, volume 8665, pages 0M 1–15, San Francisco, CA, February 5–7, 2013.

[18] Y. Qian, J. Dong, W. Wang, and T. Tan. Deep learning for steganalysis via convolutional neural networks. In A. Alattar and N. D. Memon, editors, *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2015*, volume 9409, San Francisco, CA, February 8–12, 2015.

[19] V. Sedighi, R. Cogranne, and J. Fridrich. Content-adaptive steganography by minimizing statistical detectability. *IEEE Transactions on Information Forensics and Security*, 11(2):221–234, 2016.

[20] T. Taburet, P. Bas, W. Sawaya, and J. Fridrich. A natural steganography embedding scheme dedicated to color sensors in the JPEG domain. In A. Alattar and N. D. Memon, editors, *Proceedings IS&T, Electronic Imaging, Media Watermarking, Security, and Forensics 2019*, San Francisco, CA, January 26–30, 2019.

[21] S. Wu, S.-H. Zhong, and Y. Liu. Steganalysis via deep residual network. In *IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, Wuhan, China, December 13–16, 2016.

[22] G. Xu. Deep convolutional neural network to detect J-UNIWARD. In M. Stamm, M. Kirchner, and S. Voloshynovskiy, editors, *The 5th ACM Workshop on Information Hiding and Multimedia Security*, Philadelphia, PA, June 20–22, 2017.

[23] J. Ye, J. Ni, and Y. Yi. Deep learning hierarchical representations for image steganalysis. *IEEE Transactions on Information Forensics and Security*, 12(11):2545–2557, November 2017.

[24] Y. Yousfi, J. Butora, and J. Fridrich. CNN steganalyzers leverage local embedding artifacts. In *IEEE International Workshop on Information Forensics and Security*, Montpellier, France, December 7–10, 2021.

[25] J. Zeng, S. Tan, B. Li, and J. Huang. Large-scale JPEG image steganalysis using hybrid deep-learning framework. *IEEE Transactions on Information Forensics and Security*, 13(5):1200–1214, May 2018.