Analysis and Mitigation of the False Alarms of the Reverse JPEG Compatibility Attack

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

Rémi Cogranne
cogrannr@utt.fr
University of Technology, LIST3N Lab
Troyes, France

ABSTRACT

The Reverse JPEG Compatibility Attack can be used for steganalysis of JPEG images compressed with Quality Factor 100 by detecting increased variance of decompression rounding errors. In this work, we point out the dangers associated with this attack by showing that in an uncontrolled environment, the variance can be elevated simply by using a different JPEG compressor. If not careful, the steganalyst can wrongly misclassify cover images. In order to deal with the diversity associated to the devices or softwares generating JPEGs, we propose in this paper to build a deep learning detector trained on a huge dataset of downloaded images. Experimental evaluation shows that such a detector can provide operational false alarms as small as 10^{-4}, while still correctly classifying 90% of stego images. Furthermore, it is shown that this performance is directly applicable to other image datasets. As a side product, we indicate that the attack is not applicable to images developed with a specific JPEG compressor based on the `trunc` quantization function.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → *Image compression*;

KEYWORDS

steganalysis, rounding errors, RJCA, JPEG, false alarm

ACM Reference Format:
Jan Butora, Patrick Bas, and Rémi Cogranne. 2023. Analysis and Mitigation of the False Alarms of the Reverse JPEG Compatibility Attack. In *Proceedings of the 2023 ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec '23), June 28–30, 2023, Chicago, IL, USA*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3577163.3595092

1 INTRODUCTION

In a classical steganographic scenario, Alice (the steganographer) and Bob (the receiver) are two communicating parties secretly exchanging messages through ordinary-looking media, such as digital images. There is additionally an eavesdropper, Eve, who observes their communication channel and tries to detect any suspicious activity. Alice modifies the ordinary cover images to carry a secret message, producing a stego image, in a way that is statistically undetectable by Eve. According to Kerckhoffs's principle, it is assumed that Eve knows which steganographic algorithm, and the size of the secret messages. Additionally, the *source* of images Alice is using for communication is also assumed known, where the *source* represents the parameters used to develop images. This includes a camera model, the camera settings, and possible pre-processing operations, such as downsampling, sharpening, JPEG compression, etc. Eve can therefore create her own stego images and use them to train a supervised binary detector, which can tell her whether an image from Alice is a cover or not. However, as with any binary classifier, two types of errors can occur when making a decision: a false alarm (false positive) - misclassifying a genuine cover image as a stego, or a missed detection (false negative) - erroneously classifying a stego image as an unmodified cover. These errors are often quantified by their respective probabilities for a given decision threshold, probability of false alarm P_{FA}, and probability of missed detection P_{MD}. For practical reasons[1], the operational context should focus on very small P_{FA} while, generally speaking, this is often not considered among academic works. Detectors with such "operational" false alarms can then consequently be used by law enforcement agencies for image steganalysis, even after inspecting a large number of images.

Unfortunately, steganalysis detectors suffer greatly from the so-called Cover-Source Mismatch (CSM) [10, 12, 13], which is caused by a discrepancy between the source of images used for training the detector and the source of testing images of interest [9]. Because this CSM phenomenon affects all supervised learning-based detectors, an operational false alarm can hardly be achieved. In this work, we aim to exploit a cover model of JPEG images, which is potentially general enough to be robust with respect to the CSM. This model has been introduced in the Reverse JPEG Compatibility Attack (RJCA) [5], where it is shown that decompression rounding errors of a JPEG image compressed with Quality Factor (QF) 100 follow a wrapped Gaussian distribution. More importantly, virtually any steganography transforms this distribution into a uniform one, which leads (in a controlled environment) to incredibly accurate steganalysis. However, the detectors trained in the controlled environment only provide empirical false alarm rates without any guarantees on other cover sources. While detectors with achievable

[1] A forensic analyst is usually reluctant to accuse innocent people.

IH&MMSec '23, June 28–30, 2023, Chicago, IL, USA

ACM ISBN 979-8-4007-0054-5/23/06…$15.00
https://doi.org/10.1145/3577163.3595092

theoretical bounds on false alarms have already been studied, the corresponding detectors suffer from a rather big missed detection rate [14].

In this work, we focus on the false alarm rates of machine learning detectors and the effect of the CSM on them. We point out that the assumptions made in the original RJCA work do not always hold true in an operational environment and greatly depend upon the JPEG compressor. First, we comment upon the effect of the compressor on the decompression errors and discuss their differences. We then build empirical detectors, while evaluating their false alarm rates in the presence of the CSM. We show that an operational detector providing a very small false alarm rate $P_{\mathrm{FA}} = 10^{-4}$, with a probability of (stego) detection $P_{\mathrm{D}} \sim 90\%$ can be obtained. Although the error rates are transferable to other image sources, there exists an image source, related to the `trunc` quantization function, in which the detector cannot accurately decide whether steganography has been used or not. In these cases, it is recommended to use other steganalysis methods.

The rest of the paper is organized as follows: Section 2 reminds the reader of the Reverse JPEG Compatibility Attack and introduces different JPEG compressors. In Section 3, we describe the dataset and two types of detectors used to assess the false alarm rates. The experimental results are presented in Section 4. Finally, the paper is concluded in Section 5.

2 DECOMPRESSION ERRORS

2.1 Preliminaries

Boldface symbols are reserved for matrices and vectors. Rounding x to the nearest integer will be denoted $[x]$. Similarly, $\lceil x \rceil$ and $\lfloor x \rfloor$ will denote flooring and ceiling operations. For better readability, we strictly use i, j to index pixels and k, l to index DCT coefficients. Denoting by x_{ij}, $0 \le i, j \le 7$, an 8×8 block of uncompressed, integer-valued pixels, they are first shifted by -128 to be zero-mean, a step we omit in this work as it does not have an effect on the quantities of interest. Then they are transformed during JPEG compression to DCT coefficients

$$d_{kl} = \mathrm{DCT}_{kl}(\mathbf{x}) = \sum_{i,j=0}^{7} f_{kl}^{ij} x_{ij}, \quad 0 \le k, l \le 7,$$

and quantized $c_{kl} = Q(d_{kl}/q_{kl})$, $c_{kl} \in \{-1024, \ldots, 1023\}$, where q_{kl} are the quantization steps in a luminance quantization matrix, $Q(\cdot)$ is a rounding operation, and

$$f_{kl}^{ij} = \frac{w_k w_l}{4} \cos \frac{\pi k(2i+1)}{16} \cos \frac{\pi l(2j+1)}{16},$$

$w_0 = 1/\sqrt{2}$, $w_k = 1$, $0 < k \le 7$, are the discrete cosines. The quantized DCT coefficients are then losslessly run-length encoded within every 8×8 DCT block using entropic coding.

During decompression, the above steps are reversed. For a block of quantized DCTs c_{kl}, the corresponding block of non-rounded pixels after decompression is

$$y_{ij} = \mathrm{DCT}_{ij}^{-1}(\mathbf{c} \cdot \mathbf{q}) = \sum_{k,l=0}^{7} f_{kl}^{ij} q_{kl} c_{kl}, \quad y_{ij} \in \mathbb{R}.$$

To obtain the final decompressed image, y_{ij} are rounded to integers $[y_{ij}]$. Note that typically, the integer pixels $[y_{ij}]$ are obtained straight from the unquantized DCT coefficients d_{kl} by fast integer-based inverse DCT operation.

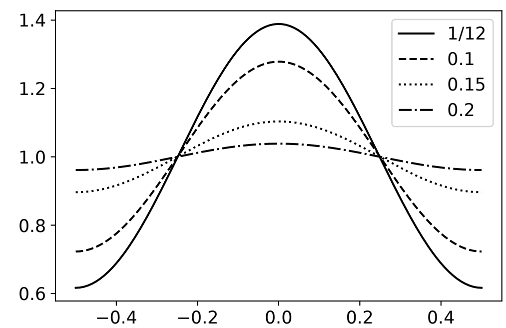

Figure 1: Probability density function of the Wrapped Gaussian distribution $\mathcal{N}_W(0, s)$ for different values of s.

The decompressed pixels can be alternatively expressed as:

$$y_{ij} = x_{ij} + \mathrm{DCT}_{ij}^{-1}(\mathbf{u} \cdot \mathbf{q}),$$

where $u_{kl} = Q(d_{kl}/q_{kl}) - d_{kl}/q_{kl}$ is the compression error in the DCT domain. The decompression rounding errors, which are the main focus of this work, are $e_{ij} = y_{ij} - [y_{ij}]$. Note that the decompression errors can now be written as :

$$e_{ij} = \mathrm{DCT}_{ij}^{-1}(\mathbf{u} \cdot \mathbf{q}) - [\mathrm{DCT}_{ij}^{-1}(\mathbf{u} \cdot \mathbf{q})],$$

therefore it is fully characterized by the compression errors u_{kl}. We will show in Section 2.3 that this error can have different properties, depending on the JPEG compressor used.

2.2 Wrapped Gaussian Distribution

Since the main goal of this paper is to assess the false alarm rate of the RJCA, we first recall in here the statistical models that were derived in the original publication. For $Y \sim \mathcal{N}(\mu, s)$ with $\mu \in \mathbb{Z}$, the rounding error $Y - [Y]$ follows a Wrapped Gaussian distribution $Y - [Y] \sim \mathcal{N}_W(0, s)$, where the probability density function (pdf) $\nu(x; s)$ of the Wrapped Gaussian is given by

$$\nu(x; s) = \frac{1}{\sqrt{2\pi s}} \sum_{n \in \mathbb{Z}} \exp\left(-\frac{(x+n)^2}{2s}\right), \tag{1}$$

with $-1/2 \le x < 1/2$. We would like to point out that the distribution parameter s represents the variance of the underlying Gaussian distribution before wrapping into interval $[-1/2, 1/2)$. If one was to compute the variance of the Wrapped Gaussian distribution, it would be smaller than the original variance s, due to the wrapping.

It was shown [5] that the rounding errors e_{ij} of a cover image follow a Wrapped Gaussian distribution

$$e_{ij} \sim \mathcal{N}_W(0, s_{ij}), \tag{2}$$

with the variance of the Gaussian distribution

$$s_{ij} = \sum_{k,l=0}^{7} (f_{kl}^{ij})^2 \mathrm{Var}(u_{kl}) q_{kl}^2. \tag{3}$$

We can see the impact of the variance on the Wrapped Gaussian distribution in Figure 1, where we show its probability density function for different variances. We see a clear evolution towards uniform distribution with increasing variances. This is important because it was also shown that the rounding errors $e_{ij}^{(S)}$ of stego images follow the Wrapped Gaussian distribution with increased variance

$$e_{ij}^{(S)} \sim \mathcal{N}_W(0, s_{ij} + r_{ij}), \tag{4}$$

where the increase of variance depends on the size of the secret message:

$$r_{ij} = \sum_{k,l=0}^{7} (f_{kl}^{ij})^2 q_{kl}^2 \beta_{kl},$$

with β_{kl} being the embedding change rates.

To derive these models, two main assumptions were made. First, the rounding errors in the DCT domain are mutually independent and follow uniform distribution between $-1/2$ and $1/2$, which is in many cases a reasonable assumption. Second, it was assumed that the embedding changes are mutually independent and also independent of the DCT rounding errors. While the second assumption depends on the steganographic scheme used and was further studied in [4], we will assume for simplicity that the embedding changes are, in fact, independent. On the other hand, we will show that the first assumption could be wrong depending on which JPEG compressor is employed.

2.3 JPEG Compressor Zoo

In this section, we introduce various JPEG compressors that will be used throughout the paper. We tried to pick the representatives of the most diverse compressors publicly available.

Convert. The first compressor we use is ImageMagick's `convert`. This compressor uses the reference libjpeg C library provided and maintained by the Independent JPEG Group (IJG) under the hood.[2] This library is provided as an open-source and therefore remains very often used in other software such as the python library PIL, Phil Sallee's Matlab JPEG toolbox, etc. This explains the amazing generalization property on these compressors previously reported in [5] (see Table V). Furthermore, Benes *et al.* [2] showed that all libjpeg versions work the same way on grayscale images compressed with QF 100. As given by the standard, `convert` uses rounding towards the nearest integer during the DCT coefficients quantization [15].

The compression error can in this case be expressed as $u_{kl} = [d_{kl}/q_{kl}] - d_{kl}/q_{kl}$ and can be therefore modeled with a uniform distribution as $u_{kl} \sim \mathcal{U}(-1/2, 1/2)$, where $\mathrm{Var}(u_{kl}) = 1/12$. It is straightforward to verify that at QF 100, the variances s_{ij} are then exactly $1/12$ for every $i, j = 0, \dots 7$, because the DCT is an orthonormal transformation and all the quantization steps are equal to 1. Note again, that s_{ij} is not the variance of e_{ij} but of the underlying Gaussian distribution before wrapping. The variance of e_{ij} can be computed numerically as 0.0638.

[2] https://ijg.org/

Mozjpeg. A very popular JPEG compressor, due to its superior compression ratio, is `mozjpeg`.[3] Not only has this compressor non-standard quantization tables (quantization is much stronger for qualities below 100), but it also by default uses Trellis quantization to help improve image quality. This is done by rate-distortion optimization on quantized DCT coefficients before the entropy coding. As a result, the quantized DCT coefficients can potentially further change their magnitude in order to use entropy codes of smaller sizes.

The compression error u_{kl} can then be modeled similarly as for `convert`. However, for a steganographic detector, the extra changes during the trellis quantization can be detected as steganographic changes with a 'small' embedding payload (4). As a result, we could expect that the variance s_{ij} will be generally bigger than for `convert`.

We want to point out, that with the trellis quantization disabled, `mozjpeg` produces the same DCT coefficients as `convert`.

Trunc. As a last compressor, we take the so-called `trunc` quantizer [1]. Instead of rounding the quantized DCT coefficients towards nearest integers, they are rounded towards zero (trunc operation - removing the fractional part): $c_{kl} = \lceil d_{kl}/q_{kl} \rceil, c_{kl} \leq 0$, $c_{kl} = \lfloor d_{kl}/q_{kl} \rfloor, c_{kl} > 0$. This truncation operation is used in various imaging devices, as it is quite efficient to implement in hardware.

The compression errors then exhibit different properties than the other compressors and can be modeled as:

$$u_{kl} \sim \begin{cases} \mathcal{U}(-1, 0), & c_{kl} > 0, \\ \mathcal{U}(0, 1), & c_{kl} < 0, \\ \mathcal{U}(-1, 1), & c_{kl} = 0. \end{cases}$$

The variance of the error in the first two cases is still $1/12$, but they are not zero-mean anymore. This is however not a problem, since the means are known and we can simply correct for them (because the quantized DCT coefficients c_{kl} are known). The problem arises when $c_{kl} = 0$ because the variance of the compression errors in these cases is $\mathrm{Var}(u_{kl}) = 1/3$. Unfortunately, even at QF 100, the majority of DCT coefficients are equal to 0, which will make the errors e_{ij} look seemingly uniform (see Figure 1).

3 BENCHMARKING SETUP

This section describes the datasets as well as the detectors used for evaluating security.

3.1 Dataset

The first dataset used for evaluation is the ALASKA2 dataset [9], which contains 80,005 uncompressed grayscale images. We JPEG compressed the whole dataset with Quality Factor 100 with several JPEG compressors introduced in Section 2.3: `mozjpeg`, `convert`, and `trunc`. We will refer to ALASKA dataset compressed with these compressors simply by their respective compressors.

For the second dataset, we downloaded 301,000 JPEG images compressed with Quality Factor 100 from Flickr[45]. and center-cropped them using `jpegtran` to 512×512 tiles. The cropping

[3] https://github.com/mozilla/mozjpeg
[4] https://www.flickr.com/
[5] According to the authors of the ALASKA competition [8], 14% of all images uploaded to Flickr have been compressed with Quality Factor 100

is done in the DCT domain, thus avoiding recompression. In this dataset, we do not know anything about the compressor used. After a very brief inspection of DCT histograms, it even seems that some images might have been double-compressed. We are hoping that training a detector on this diverse dataset will provide it with enough generalization power to other JPEG compressors used. Note that this forces us to (blindly) annotate all the images from this dataset as cover images.

To create stego images, we embed the Flickr and `convert` datasets with UERD [11] at payload 0.4 bits per non-zero AC DCT coefficient (bpnzac). This payload is big enough to change completely the cover statistics and since we are mainly interested in the false alarm rate, we can afford not detecting some stego images with smaller payloads.

For training the deep learning detectors, we split the `convert` dataset into training, validation, and testing set of sizes 66k, 4k, and 10k images respectively. In order to have a reliable estimate of small false alarm (e.g. 10^{-4}) in the bigger Flickr dataset, we split it into training, validation, and testing sets of 146k, 10k, and 145k images. We chose such a big testing set on purpose, in order to have reliable estimates of false alarms as small as 10^{-4}.

3.2 Detectors

For experimental evaluation, we use two types of detectors in this work. The first detector is a variance detector using the variance of decompression rounding errors

$$\mathrm{V} = \frac{1}{N}\sum_{i=1}^{N} e_i^2 \tag{5}$$

as a test statistic, where N is the number of pixels in the image. The detector is then tuned by establishing a threshold λ, such that the image is classified as a cover image if $\mathrm{V} < \lambda$, and is classified as a stego otherwise.

For the second detector, we chose the state-of-the-art e-SRNet [5], which is equivalent to SRNet [3] trained on the decompression errors e_{ij}. The detector was trained with a mini-batch size of 32 images, weight decay 2×10^{-4}, one-cycle learning rate with maximal value at 10^{-3}, and Adamax optimizer. The training was set for 10 epochs in the Flickr dataset and 20 epochs for ALASKA2, due to its smaller training set.

3.3 Error Filtering

Because the assumptions on decompression errors do not always hold, we will in our investigation filter out all 8×8 blocks that do not follow these assumptions. These are, as far as we are aware, blocks with *near-constant* content [7]. In this work, we consider a block to be near-constant if the variance of its pixels is below 2. Let $\mathcal{I}$ be the set of all pixels from blocks that are not near-constant. The filtered variance is then computed as

$$\mathrm{FV} = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} e_i^2. \tag{6}$$

Figure 2 shows a comparison between ROC curves of the variance detectors using the original decompression error variance V and the filtered variances FV. We see a clear improvement in the curve by employing the proposed filtering. To this end, unless stated

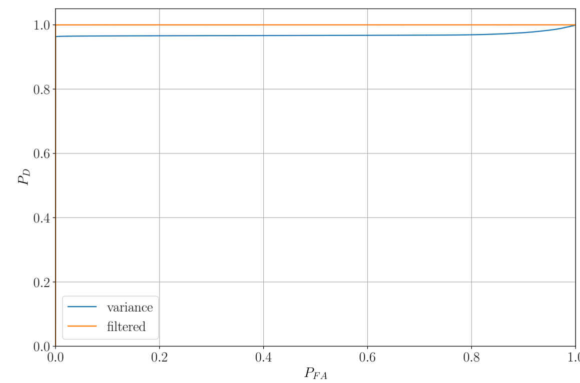

Figure 2: ROC curve of a variance detector with and without filtering.

otherwise, we will always consider the block filtering. On the other hand, the deep learning detector will not preprocess the images in this way, as we believe it is not necessary.

In Figure 3 we show histograms of the filtered variances FV. We can observe what we briefly discussed above. The ALASKA2 images compressed with `convert` have in general smallest variances, `mozjpeg` increases the variances slightly and `trunc` produces variances close to 1/12. For the Flickr images, we can observe that the density is multimodal, which by itself suggests that very different compressors (possibly combined with other image processing operations) are present in this dataset. We leave the analysis for future research but believe that these outliers are linked to the rounding operations that occur in digital cameras and that are used in image forensics by analyzing dimples [1]. Finally, we can see that embedding images with UERD [11] at 0.4 bpnzac also increases variance, as given by Equation (4).

4 RESULTS

In the following, we discuss the strategies for training the detectors and comment upon their results. For both detectors, we first inspect the ROC curves on ALASKA2 compressed with `convert` and Flickr datasets, both embedded with UERD at 0.4 bpnzac. Next, we will discuss the false alarm rates of these detectors on cover images coming from the other source.

4.1 Variance Detector

First, we will use the variance detector tuned on `convert` dataset. Even though this detector uses only the variances FV, Figure 4 shows that, even with a false alarm rate of 10^{-4}, perfect detection of UERD (probability of detection $P_{\mathrm{D}} = 1$) is achievable. Note that since the dataset has only 80k images, the results for smaller false alarms are rather noisy. Unfortunately, if we use this detector on other image sources, the same figure shows that the false alarms increase drastically (with the exception of `mozjpeg` at FA 10^{-4}). We hypothesize that this is due to the limited variability of the JPEG compressor in the data used for tuning the detector.

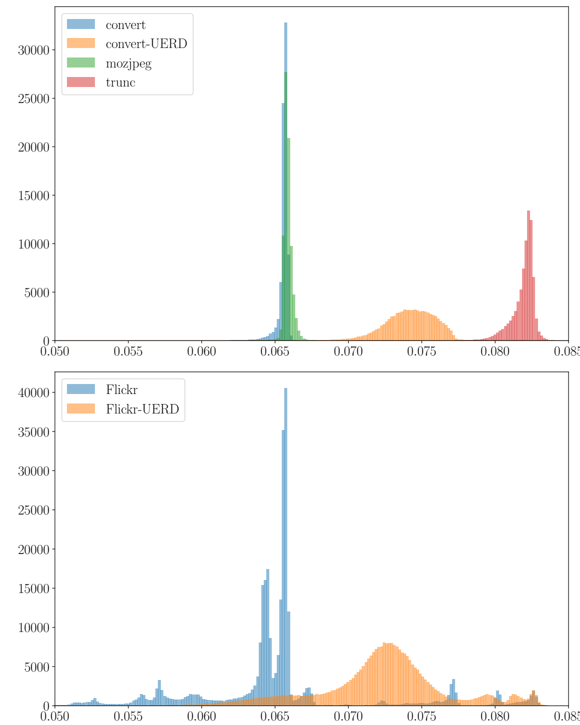

Figure 3: Histograms of the filtered variances FV **for different classes of images. Top: ALASKA2, Bottom: Flickr.**

We, therefore, repeat the experiment with the variance detector, this time tuned on the Flickr dataset instead. In Figure 5, we see that the false alarms below 10^{-3} are now much more nicely behaved, that is they are not greater than what we prescribe in the Flickr dataset.[6] However, after inspecting the top curves, we see that by introducing a more diverse cover source, we sacrificed almost all the detection power of the detector. This is even more obvious from Table 1, where we show P_{D} in the training source and P_{FA} in the other datasets. We conclude that a more complex detector needs to be used.

4.2 Deep Learning

We now extend the experiments with the variance detector to a more advanced detector, the e-SRNet. As previously, we first train the detector on `convert` embedded with UERD at 0.4 bpnzac. It will not come as a surprise, that the detector also achieves a perfect ROC curve, see Figure 6. Nonetheless, we can also observe that testing other cover images produces false alarms even worse than with the variance detector. We believe this happens because the detector is complex enough to overfit the given JPEG compressor.

We thus try to use the detector's complexity to contain information about as many JPEG compressors as possible by training on part of the Flickr dataset.[7] While we can observe a small drop in

[6] Note that this last feature is important for decision-making purposes: the forensic analyst will be able to adopt a conservative decision on detecting stego contents if the practical error rate is known to be equal to or lower than the prescribed one.
[7] Clearly not all possible JPEG compressors are used but we believe that the Flickr dataset contains large enough samples of the most popular compressors.

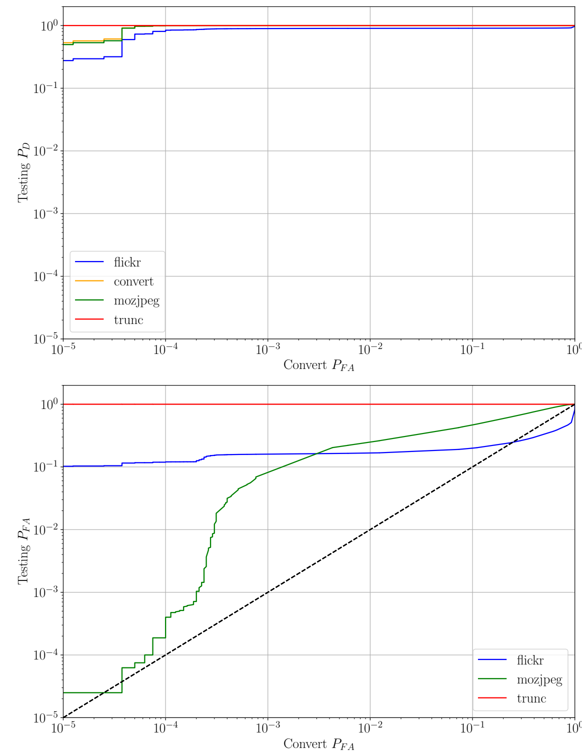

Figure 4: Variance detector tuned on `convert`. P_{D} (top) and P_{FA} (bottom) in other sources as a function of P_{FA} in `convert`.

Train	P_{D}	P_{FA}			
		`convert`	Flickr	`mozjpeg`	`trunc`
`convert`	0.9636	10^{-1}	0.2026	0.4826	0.9999
	0.9635	10^{-2}	0.1678	0.2617	0.9999
	0.9634	10^{-3}	0.1637	0.2030	0.9999
	0.9629	10^{-4}	0.1190	0.0002	0.9999
Flickr	0.2447	0	10^{-1}	0	0.9997
	0.0108	0	10^{-2}	0	0.0826
	0.0009	0	10^{-3}	0	0.0017
	0.0001	0	10^{-4}	0	0

Table 1: Variance detector cross-testing of false alarms on different JPEG compressors. Each row corresponds to a detector with a fixed threshold.

detection for very small false alarms (see Figure 7), the generalization capabilities on other cover datasets are more than satisfying - for every other dataset, if the prescribed false alarm rate is below 10^{-3}, then the false alarm on other datasets is also bounded by this prescribed value. Based on these observations, we conclude that the Flickr dataset contains (among others) images compressed with all compressors studied in this work: `convert`, `mozjpeg`, and `trunc`. Since the detector is complex enough, we can see in Table 2 that even for a very conservative false alarm of 10^{-4}, we still achieve 87% detection on the Flickr dataset. However, a problem

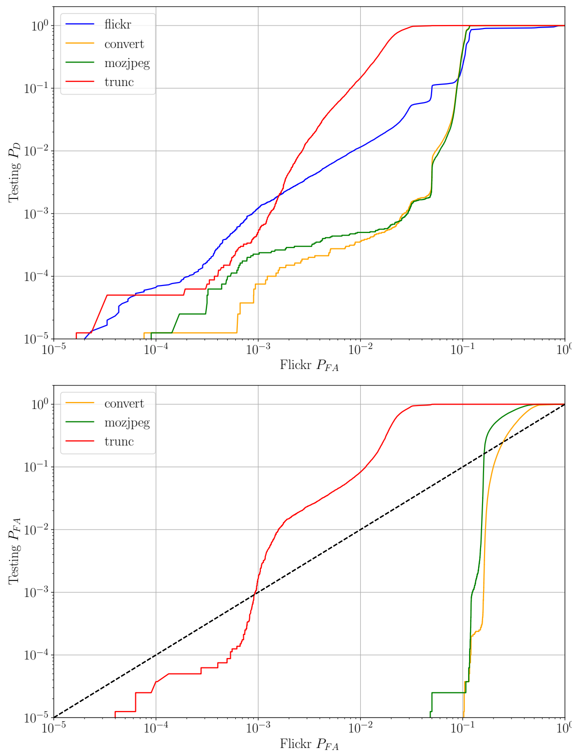

Figure 5: Variance detector tuned on Flickr. P_{D} (top) and P_{FA} (bottom) in other sources as a function of P_{FA} in Flickr.

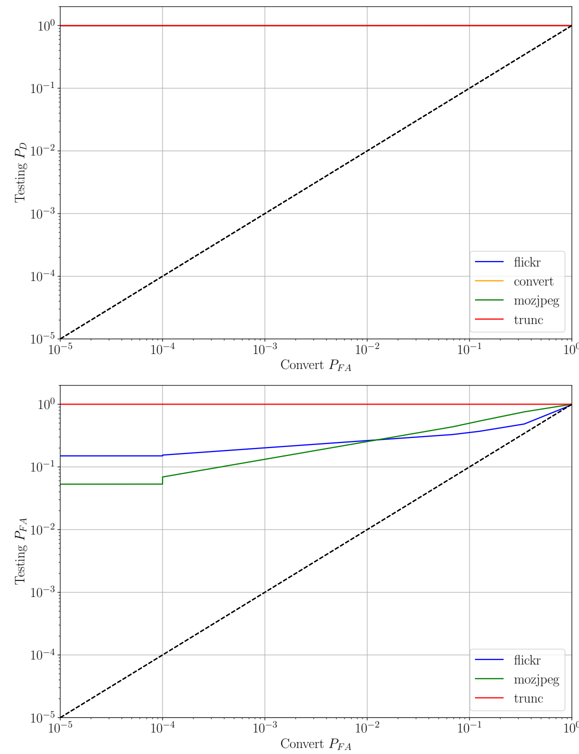

Figure 6: e-SRNet trained on `convert`. P_{D} (top) and P_{FA} (bottom) in other sources as a function of P_{FA} in `convert`.

Train	P_{D}	P_{FA}			
		`convert`	Flickr	`mozjpeg`	`trunc`
`convert`	1	10^{-1}	0.3527	0.4968	1
	1	10^{-2}	0.2498	0.2578	1
	1	10^{-3}	0.1805	0.1507	1
	1	10^{-4}	0.1576	0.0831	0.9999
Flickr	0.9999	0.0161	10^{-1}	0.1403	1
	0.9751	0	10^{-2}	0.0005	0.3865
	0.9602	0	10^{-3}	0.0002	0.0009
	0.8737	0	10^{-4}	0	0
Flickr -filtered	0.9999	0.0473	10^{-1}	0.1863	N/A
	0.9999	0.0013	10^{-2}	0.0200	N/A
	0.9999	0	10^{-3}	0.0013	N/A
	0.9264	0	10^{-4}	0	N/A

Table 2: e-SRNet cross-testing of false alarms on different JPEG compressors. Each row corresponds to a detector with a fixed threshold.

arises after inspecting what happens with stego images in the other sources. While we get a reasonable detection in Flickr, `convert`, and `mozjpeg` ($\sim 85\%$ for $P_{\mathrm{FA}} = 10^{-4}$), the accuracy on stego images in the `trunc` set is 0%. To investigate why all stego images from this source are treated as covers, we investigate the detector's logits of the stego class.

Unidentifiable Images. As mentioned above, we investigate here the soft outputs (logits) of the e-SRNet trained in the Flickr dataset (see Figure 7 and Table 2). We collect the logits from all 4 sources (only test set images) and show histograms of the corresponding logits (cover and stego images) in Figure 8. We can make several observations from this figure. First, we see that in the Flickr dataset, we have a reasonable separation of cover and stego classes, except from the middle lobe around zero containing both classes. Not surprisingly, for `convert` and `mozjpeg`, we get perfect separation. We can note that the distribution of `mozjpeg` cover images has a thicker right tail, which we attribute to the trellis quantization. Lastly, we see that the detector is randomly guessing in the `trunc` source. This is also not so surprising based on the rounding error analysis from Section 2.3. Unfortunately, this means that the e-SRNet is not applicable to steganalysis in the `trunc` source, because the statistic of interest looks like a uniform noise for both cover and stego images. While there are alternative methods for the steganalysis of these images [6] (such as detection in the pixel or DCT domain), it is not the goal of this work.

In order to avoid having a detector that always blindly assigns a cover class, we instead modify our already established detector to restrain from decisions on such images. To do this, we first observed that the largest logit coming from the `trunc` images is 1.91. We then set up a threshold $T = 2$ and force the detector to discard all images whose logit is in absolute value smaller than T. In practice, the steganalyst would need to use another detector to

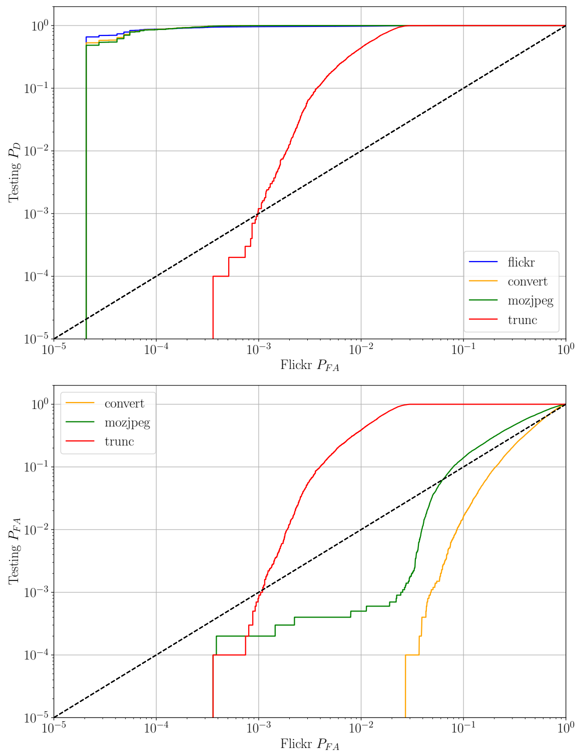

Figure 7: e-SRNet trained on Flickr. P_{D} (top) and P_{FA} (bottom) in other sources as a function of P_{FA} in Flickr.

make a decision in these images. In Figure 9, we show the false alarms in different cover sources of this new detector with such filtering. We see that the other sources now follow the prescribed false alarm much more accurately, with the exception of the `trunc` images that have been all filtered out, thus the detector cannot decide on them, see Table 2 for specific values of false alarms. We now fixed the detector's decision threshold for $P_{\mathrm{FA}} = 10^{-4}$ and show in Table 3 the false alarms in other sources as well as the detection of steganography. We conclude that the proposed filtering on logits increased detection accuracy in Flickr by 5%, while not affecting `convert` and `mozjpeg` sources. Note also that the false alarm in `convert` and `mozjpeg` is technically not zero, but due to smaller testing dataset size, we cannot reliable estimate values below 10^{-4}. It is also shown that the filtering procedure discards 5.7% of all images from Flickr, 1.3% from `convert` and 3.6% from `mozjpeg`. Also, by design, all images from `trunc` are discarded from the decision-making.

5 CONCLUSIONS

In this work, we studied the cover source impact on the Reverse JPEG Compatibility Attack. The main power of the attack comes from an increased variance of the decompression rounding errors. We thus study these rounding errors in several popular JPEG compressors. We have shown that the variance of these rounding errors can also change drastically with different compressors, potentially triggering a lot of false alarms. Indeed, we showed that a naive

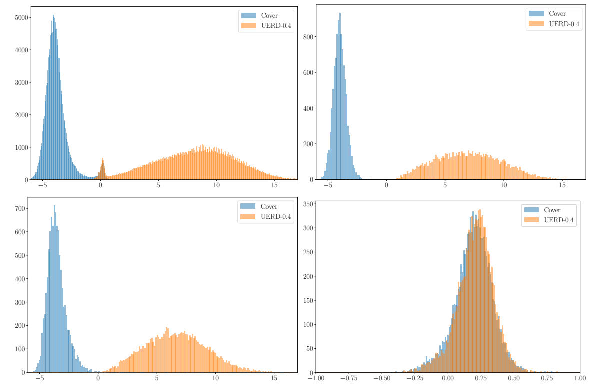

Figure 8: Histogram of logits (stego class) of e-SRNet trained on Flickr. From left to right: Flickr, `convert`, `mozjpeg`, `trunc`.

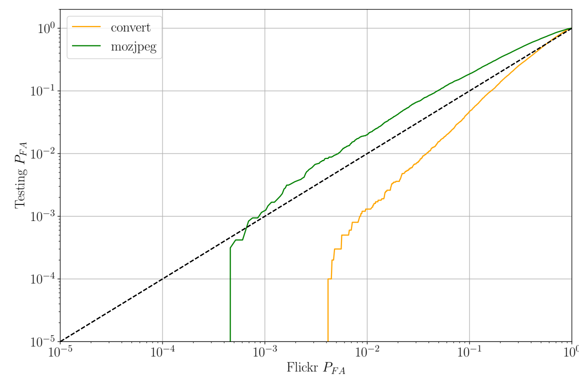

Figure 9: P_{FA} in other sources as a function of P_{FA} in Flickr of e-SRNet with the logit filtering.

	Flickr	`convert`	`mozjpeg`	`trunc`
P_{FA}	10^{-4}	0	0	N/A
P_{D}	0.9264	0.8841	0.8781	N/A
unclassified	0.0574	0.0137	0.0364	1

Table 3: Probability of detection of UERD at 0.4 bpnzac with the filtered e-SRNet trained on the Flickr dataset. Threshold for false alarm rate 10^{-4} in Flickr was used.

variance detector does not generalize on cover images compressed differently. Using it in a more diverse source, on the other hand, makes the detector classify even the stego images as covers. A more complex deep learning detector trained on a diverse enough dataset preserves false alarms across cover sources, but we point out that the `trunc` quantization still makes the rounding errors unusable for steganalysis. However, we show that in a vast majority of cases, it is possible to achieve operational false alarm rates while still having very high detection power even on datasets not included in the training data. In the future, we plan to investigate the images in which the detector does not provide a confident decision by inspecting their respective EXIF data.

ACKNOWLEDGEMENT

This work was granted access to the HPC resources of IDRIS under the allocation 2022-AD011012855 made by GENCI. This work received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 101021687 (project "UNCOVER").

REFERENCES

[1] S. Agarwal and H. Farid. Photo forensics from rounding artifacts. In C. Riess and F. Schirrmacher, editors, *The 8th ACM Workshop on Information Hiding and Multimedia Security*, Denver, CO, 2020. ACM Press.

[2] M. Beneš, N. Hofer, and R. Böhme. Know your library: How the libjpeg version influences compression and decompression results. In *The 10th ACM Workshop on Information Hiding and Multimedia Security*, Santa Barbara, CA, June 27–29, 2022. ACM Press.

[3] 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.

[4] J. Butora and P. Bas. Fighting the reverse JPEG compatibility attack: Pick your side. In *The 10th ACM Workshop on Information Hiding and Multimedia Security*, Santa Barbara, CA, June 27–29, 2022. ACM Press.

[5] J. Butora and J. Fridrich. Reverse JPEG compatibility attack. *IEEE Transactions on Information Forensics and Security*, 15:1444–1454, 2020.

[6] J. Butora and J. Fridrich. Steganography and its detection in JPEG images obtained with the "trunc" quantizer. In *Proceedings IEEE, International Conference on Acoustics, Speech, and Signal Processing*, Barcelona, Spain, May 4–8, 2020.

[7] R. Cogranne. Selection-channel-aware reverse JPEG compatibility for highly reliable steganalysis of JPEG images. In *Proceedings IEEE, International Conference on Acoustics, Speech, and Signal Processing*, pages 2772–2776, Barcelona, Spain, May 4–8, 2020.

[8] R. Cogranne, Q. Giboulot, and P. Bas. The ALASKA steganalysis challenge: A first step towards steganalysis "Into the wild". In R. Cogranne and L. Verdoliva, editors, *The 7th ACM Workshop on Information Hiding and Multimedia Security*, Paris, France, July 3–5, 2019. ACM Press.

[9] R. Cogranne, Q. Giboulot, and P. Bas. ALASKA–2: Challenging academic research on steganalysis with realistic images. In *IEEE International Workshop on Information Forensics and Security*, New York, NY, December 6–11, 2020.

[10] Q. Giboulot, R. Cogranne, D. Borghys, and P. Bas. Effects and Solutions of Cover-Source Mismatch in Image Steganalysis. *Signal Processing: Image Communication*, August 2020.

[11] L. Guo, J. Ni, W. Su, C. Tang, and Y. Shi. Using statistical image model for JPEG steganography: Uniform embedding revisited. *IEEE Transactions on Information Forensics and Security*, 10(12):2669–2680, 2015.

[12] A. D. Ker and T. Pevný. A mishmash of methods for mitigating the model mismatch mess. In A. Alattar, N. D. Memon, and C. Heitzenrater, editors, *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2014*, volume 9028, pages 1601–1615, San Francisco, CA, February 3–5, 2014.

[13] J. Kodovský, V. Sedighi, and J. Fridrich. Study of cover source mismatch in steganalysis and ways to mitigate its impact. In A. M. Alattar, N. D. Memon, and C. D. Heitzenrater, editors, *Media Watermarking, Security, and Forensics 2014*, volume 9028, pages 204 – 215. International Society for Optics and Photonics, SPIE, 2014.

[14] Etienne Levecque, John Klein, Patrick Bas, and Jan Butora. Toward reliable jpeg steganalysis (at qf100). In *2022 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2022.

[15] W. Pennebaker and J. Mitchell. *JPEG: Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, 1993.