https://doi.org/10.2352/ISSN.2470-1173.2020.4.MWSF-076

Detection of Malicious Spatial-Domain Steganography over Noisy Channels Using Convolutional Neural Networks

Swaroop Shankar Prasad, University of Stuttgart, Germany, swaroopsprasad@gmail.com; Ofer Hadar, Ben-Gurion University of the Negev, Israel, hadar@bgu.ac.il; Ilia Polian, University of Stuttgart, Germany, ilia.polian@informatik.uni-stuttgart.de

Abstract

Image steganography can have legitimate uses, for example, augmenting an image with a watermark for copyright reasons, but can also be utilized for malicious purposes. We investigate the detection of malicious steganography using neural network-based classification when images are transmitted through a noisy channel. Noise makes detection harder because the classifier must not only detect perturbations in the image but also decide whether they are due to the malicious steganographic modifications or due to natural noise. Our results show that reliable detection is possible even for state-of-the-art steganographic algorithms that insert stego bits not affecting an image's visual quality. The detection accuracy is high (above 85%) if the payload, or the amount of the steganographic content in an image, exceeds a certain threshold. At the same time, noise critically affects the steganographic information being transmitted, both through desynchronization (destruction of information which bits of the image contain steganographic information) and by flipping these bits themselves. This will force the adversary to use a redundant encoding with a substantial number of error-correction bits for reliable transmission, making detection feasible even for small payloads.

Keywords: *Image steganography, Malicious steganography, Steganalysis, Transmission over noisy channels, Deep Learning, Convolutional Neural Networks*

Introduction

Image steganography [1] refers to embedding of information ("payload") P into an image I in a manner that is difficult to notice by a human viewer (see Fig. 2 in the next section). Image steganography can have legitimate or malicious (adversarial) purposes. Among legitimate uses of steganography are watermarking, e.g., adding information related to the image's author to protect his/her intellectual property rights, and authenticating images to counteract neural-network based "deepfake" attacks [2]. However, this paper deals with undesired, malicious steganography.

Fig. 1 illustrates two scenarios where communication over a steganographic channel is used maliciously. In Fig. 1a, Alice leaks sensitive data (such as private images not intended for Bob or cryptographic key material) from the protected system, and Bob receives them without legitimacy. This information leakage can be considered a passive threat. In contrast, an active threat involving the same parties is shown in Fig. 1b: Bob uses the steganographic channel to control malware previously planted into the protected system. The purpose of this work is to detect malicious steganography using neural networks. Consistent with the usual naming in cryptography, Fig. 1 refers to this role as "Eve". Note that, in contrast to classical encryption scenario, here Alice and Bob are trying to violate and Eve is trying to defend the system's security.

In this paper, we consider the problem of detecting malicious steganography when images are transmitted over a noisy channel, such that some of the pixels of the received image do not exactly match the original (sent) image. Examples of applications affected by noise are low-power surveillance systems that communicate using wireless links with signals of marginal strength, and monitors of (environmental) parameters that transmit images over repeaters placed in potentially noisy environments. We assume that the communication parties (Alice and Bob in Fig. 1) "hijack" an existing transmission channel over which images are being sent (and the quality loss due to noise is tolerable for the specific application). Alice and Bob can modify the images to embed stego information, but they cannot improve the channel, e.g., make it retransmit an image several times, add error-correcting information or increase the signal strength.

The noisy channels complicates the situation of both: the attackers (Alice and Bob) and the defender (the detection procedure, or Eve in Fig. 1). From the attacker's point of view, not all information encoded in the stego bits arrives. For example, if Alice is trying to leak a password to Bob, Bob will receive a distorted bit string that will not let him log in. The distorted password may still be of use for Bob (e.g., he may apply single or

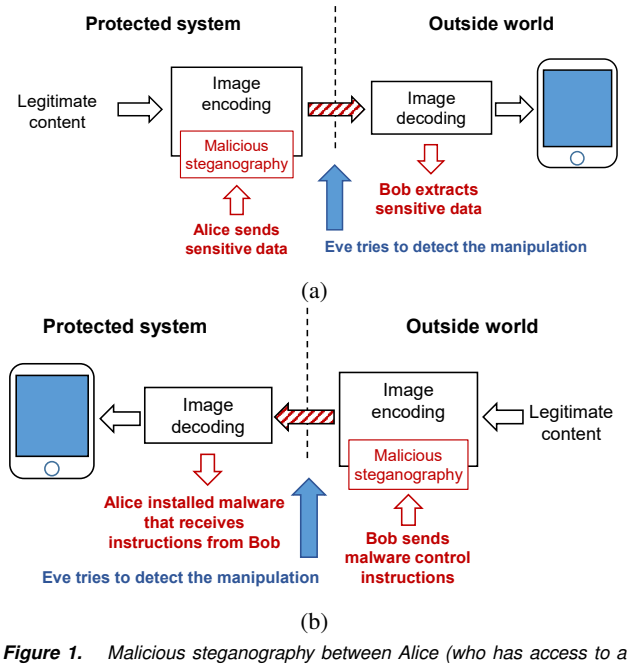
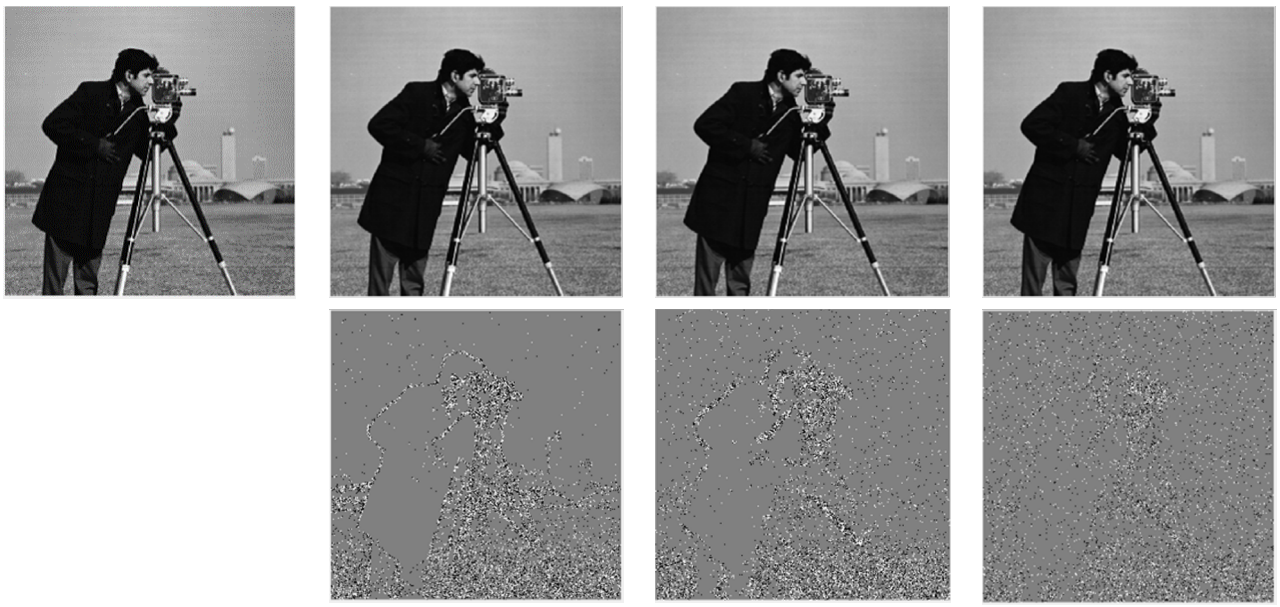


Figure 1. *Malicious steganography between Alice (who has access to a protected system but no communication channel to outside world) and Bob (located in the outside world).*



(a) (b) (c) (d)

Figure 2. *(a) Example original image and image embedding steganographic information for (b) HUGO [3], (c) WOW [4] and (d) S-UNIWARD [5]. The bitmaps below show the differences to the original image due to stego bits.*

double bit-flips to the received data and try to log in with obtained data); while this is a far better procedure than brute-force search, many systems will deny access after a few attempts, rendering the attack useless. In the active scenario in Fig. 1b, the malware can be extremely resource constrained to avoid being discovered and may not have the capabilities to do fault-tolerant decoding of instructions received through the noise-affected stego channel.

From the defender's point of view, the classification procedure must distinguish the (undesired) effects of steganography from the (unavoidable) consequences of noise. We are using a convolutional neural network from an earlier publication [6] for classification. In contrast to this (and other existing works on deep-learning based steganalysis [7, 8, 9, 10]), we train the network using cover and stego images affected by different types and intensities of noise. We are investigating both: the impact of noise on the steganographic channel (using three state-of-the-art spatial-domain steganographic algorithms), and on the detection performance of the neural network.

The remainder of this paper is organized as follows. The considered steganographic methods and the noise models are explained in the next section. The convolutional neural network and the learning procedures employed for detecting malicious steganography are discussed afterwards. Finally, experimental results are reported.

Noise-affected Malicious Steganography

We consider popular *spatial-domain stego algorithms* WOW [4], HUGO [3, 11] and S-UNIWARD [5] (see Fig. 2). Steganographic methods in transform domain [12] or video steganography based on motion estimation [13] are not in scope of our work.

Spatial-domain steganography

In spatial-domain steganography [14], an image I is interpreted as an array of bits $I_1,\ldots,I_n$ (e.g., $n = 512\times512\times8$ bits for a 512×512 pixel greyscale image). A *steganographic algorithm* $\mathscr{S}$ takes the cover image I and a bit-string P called *payload* and produces the *stego image* $I_S := \mathscr{S}(I,P)$. In *fixed-key steganographic algorithms*, the subset $M \in \{1,\ldots,n\}$ of I's bits is agreed in advance between the communicating parties, and $\mathscr{S}$ could just write P's bits into positions from M; here, M serves as the steganographic key. A simple fixed-key stego algorithm could use the LSBs of every 16-th pixel in I as stego bits.

From the adversary's point of view, it is more attractive to use different keys for different images, taking into account knowledge about the image, e.g., hiding stego bits near to sharp edges or other high-activity regions within the image where they are difficult to recognize. However, transmitting an individual key for each image requires a separate communication channel, which does not exist in malicious steganography (if Alice and Bob of Fig. 1 had a channel to send the stego bit positions, they could use the same channel to leak information or control the malware without any need for steganography in the first place).

State-of-the-art steganographic algorithms used in this paper follow an intermediate path: they create a pseudo-random matrix `RandChange` of size equal to the image and use this matrix as a (fixed) key. Both the sender and the receiver know `RandChange` (it is sufficient to share the seed of the pseudorandom number generator used to create this matrix). Then, given a cover image, an algorithm computes, for each image pixel, a *cost function* (e.g., the proximity of the pixel to a sharp edge) and decide whether to use the pixel for stego data or not based on the comparison of the calculated cost function with the corresponding position of `RandChange`. Therefore, even though the same key is used, stego bits are inserted at different, image-dependent positions.

Modeling and impact of noise

We assume that the stego image is transmitted through a channel affected by two kinds of noise: Gaussian noise applied to each pixel and packet loss, modeled as a burst of 2×2, 4×4, 8×8 or 16×16 pixel blocks being replaced by values 0 (black). Examples of both kinds of noise can be found in Fig. 4. This transmission transforms an image—with or without steganographic information—into a distorted image. We write the distorted versions of the original and the stego image as I' and I'_S, respectively.

Note that we assume natural noise, not human intervention with the image, like cropping or resizing (which would also be easy to detect). It is possible to eliminate or reduce noise on the communication protocol level, e.g., via re-transmissions or using error-correcting codes [15, 16, 17], but we assume scenarios where such techniques either have not been applied or do not fully eliminate all occurring noise.

From the attacker's point of view, the noisy channel has two effects: bit-flips on the stego bits and desynchronizing the stego algorithm. The first effect occurs for each algorithm $\mathscr{S}$: wherever the stego bits are located, they are subject to the same noise as all other bits of the image and can change their value (flip). *Desynchronization* can occur for complex stego algorithms which determine the stego bits positions based on an image's features, when these very features were modified during transmission due to noise. For example, assume that algorithm $\mathscr{S}$ selected stego bit positions based on sharp edges in I; the payload was embedded and the stego image transmitted. A few packets were lost during transmission and the received image contains black pixel blocks that were not present in the original image. When the receiving party re-runs $\mathscr{S}$ to determine stego bit positions, different positions (and in general a different number of them) will be returned; the bits on these positions will not contain the stego data.

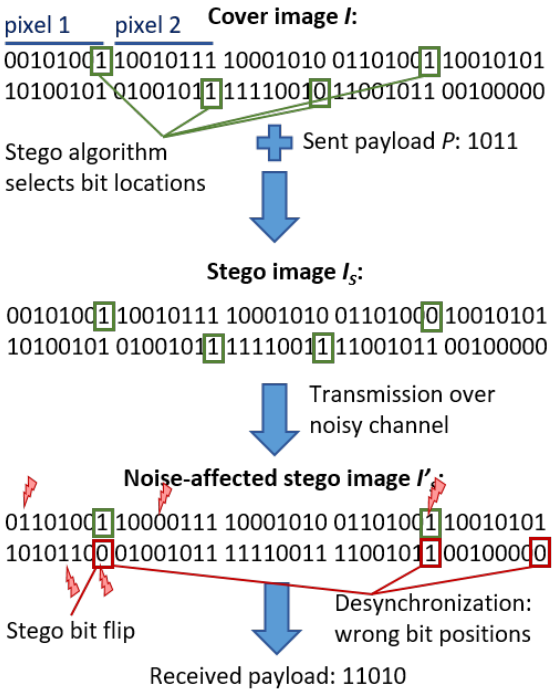


Figure 3. *Example of distortions during transmission of a payload through the steganographic channel*

Fig. 3 illustrates the distortions. Algorithm $\mathscr{S}$ selects four locations within the image data I, and the bits of payload $P = 1011$ are written into these locations, resulting in image I_S. During the transmission, five bits, including one stego bit, are flipped. Running $\mathscr{S}$ on the received distorted image I'_S results in desynchronization: five instead of four locations are identified, only two of them correct. The received payload 11010 is completely unrelated to the sent payload P.

The adversary can try to counteract noise by applying redundancy, e.g., transmitting the same information multiple times or use error-correcting codes [15, 16, 17]. This will decrease the bandwidth of the steganographic channel, and the adversary will have to use more images to store the same stego information, thus increasing the chances of being detected. As an alternative, the attacker can increase the number of payload bits encoded into one image (the used stego algorithms provide parameter α that supports such a decision). As we will see later, this increases the probability of detection as well.

Desynchronization is more difficult to counteract, because the sending party (e.g., Alice in Fig. 1) does not know which and how many bits are lost. Even worse, malicious steganographic channels are usually unidirectional. Even if Alice included, e.g., synchronization sequences or checksums into the stego message, Bob has no chance to acknowledge a correct receipt of the message or request a re-transmission. We are not aware of a satisfactory solution to the desynchronization problem and expect that if it exists, it comes with a heavy overhead. Without such a solution, the adversary must resort to simple, non-adaptive stego algorithms that store stego bits in fixed positions; such manipulations should be easier to detect than the sophisticated algorithms considered in this paper.

Steganography Detection

The detection (or steganalysis) problem addressed in this paper is: given a received distorted image I', decide, using deep learning, whether the image contains steganographic manipulations or not, without knowing the undistorted images I or I_S. In contrast to extensive literature on image steganalysis [7, 8, 9, 10], we focus on images received through noisy channels.

We adopt the neural network initially introduced in [6]. The network consists of 2 Convolutional layers with hyperbolic tangent activation function. In the first convolutional layer a single kernel of size 3×3 is used, whereas 64 kernels each of size 509×509 is employed in the second layer. No Pooling layer is used and the fully connected part is a simple output layer with 2 softmax neurons. Stochastic Gradient Descent (SGD) optimization algorithm with a batch size of 100 samples, learning rate of 0.005, decay of 5e-7 and zero momentum is chosen. In contrast to [6], we applied early stopping with a patience of 5 to overcome overfitting and improve detection accuracy. The neural network was modelled using TensorFlow 2.0. All images were normalized to [0,1] before feeding it into the neural network.

Experimental Results

We perform two sets of experiments: we investigate the impact of noise on the error rate of the steganographic channel and on the detection performance of the neural network from the last section.

Figure 4. *Example effects of Gaussian noise (middle) and packet loss (right)*

Generation of the dataset

We use images from the BOSSBASE [18] and RAISE [19] datasets. More specifically, the training set is constructed from first 7,000 images from BOSSBASE and first 4,000 from RAISE; the test set from last 3,000 images from BOSSBASE and next 1,000 from RAISE; and the prediction set from 2,000 of remaining 3,156 images from RAISE. Since each image is used in one stego-free and one stego version, the total cardinalities of the three sets are 22,000 (training), 8,000 (test) and 4,000 (prediction). We used HUGO, WOW and S-UNIWARD stego algorithms with fixed-stego key implementations from [20].

Fig. 5 summarizes how the sets of distorted images I' and I'_S used for experiments are obtained. For each image, we can decide which of the three considered stego algorithms will be used and set parameter α. (Essentially, α is the proportion of stego bit locations found by the stego algorithm that will be used; for instance, $\alpha = 0.4$ means that payload bits will be placed on 40% of the identified locations). Both images (the cover image I and the stego image I_S) are then distorted by adding Gaussian noise and packet loss. We aim at images which have a sufficient quality, quantified by distortion-related PSNR between 30 and 60 (a noisy channel that leads to a lower PSNR would perhaps be rather useless in practice). Gaussian noise is determined by its standard deviation; we use values between 0.3 and 11 because they lead to PSNR of 60 and 30, respectively. Packet loss is defined by the size of the block that is zeroed when one packet is lost (we considered 2×2, 4×4, 8×8 or 16×16 pixel blocks); the *packet loss ratio* PLR (values between 0.001 and 0.01) and the *average burst length* ABL (values between 4 and 30) [21].

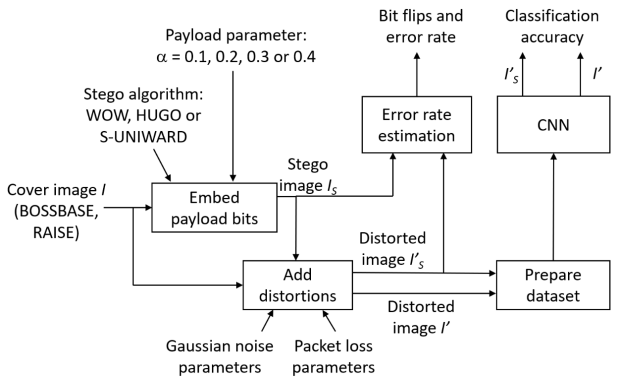


Figure 5. *Generation of experimental data*

Error-rate estimation

Table 1 summarizes our results on the impact of the noisy channel on steganographic operation for the HUGO algorithm. We observed that for adaptive algorithms used here, the transmission of an image over an unreliable channel practically always results in desynchronization, i.e., different numbers and positions of stego bits found before and after transmission. This suggests that a steganographic algorithm should include synchronization features, even though their optimal design is currently unclear. Table 1 shows the effect of various types of noise on the error rate of the steganographic channel assuming perfect synchronization (the receiver knows the correct bit positions).

Each row of Table 1 corresponds to a combination of noise sources with specific parameters. For packet loss, these are the packet size (block in the image that is zeroed if a packet is lost), ABL and PLR explained above; we also report how many packets with the selected size are included in the 512×512 pixel image and how many of them were lost. For the Gaussian noise, we consider two values 0.3 and 11 (which lead to a PSNR of roughly 60 and roughly 30, respectively). We run HUGO algorithm with two representative values of α and report the stego bits accommodated within the image. For each set of noise parameters and each α, Table 1 includes the PSNR of the distorted image, the number of bit flips affecting the stego bits in this image, and the error rate (the latter number divided by the total number of stego bits).

Interestingly, the error rate is largely determined by the standard deviation of the Gaussian noise and practically unaffected by parameters related to packet loss. This is because even very few lost pixel blocks dramatically deteriorate the image quality; for instance, already nine 8×8 blocks are sufficient to reach our minimal target PSNR around 30. Relatively few stego bits are affected in such images, and the error rate seen by the adversary is low. In contrast, Gaussian noise worsens the quality gradually, and relatively many bits—including stego bits—must be affected for PSNR between 30 and 60. We observe rates between 5% and 25% that can be considered severe and call for a strong error-correcting solution. Even if the attacker is transmitting short pieces of information (e.g., 128-bit encryption keys or control instructions for malware), a large number of redundant bits will be needed to communicate them reliably.

Space constraints prevent us from publishing the counterparts of Table 1 for the other two stego algorithms, but the results are almost algorithm-independent. Fig. 6 summarizes the average

Table 1: Error rate over steganographic channel for HUGO

Noise Parameters							α = 0.4 (23,523 stego bits)			α = 0.1 (4,717 stego bits)		
Noise Type	Packet Size	ABL	PLR	Total Packets	Packets lost (avg.)	Std. dev	Avg. PSNR	Avg. bit flips	Error rate[%]	Avg. PSNR	Avg. bit flips	Error rate [%]
Gaussian only	-					0.3	56.8	1111	4.73	60.07	223	4.73
						11	30.59	5907	25.11	30.59	1177	24.96
Packet Loss only	2×2	11	0.0035	65535	221	-	32.29	40	0.17	32.29	8	0.17
Gaussian + Packet Loss						0.3	32.28	1164	4.95	32.28	229	4.85
						11	28.38	5872	24.97	28.38	1181	25.03
Packet Loss only	4×4	21	0.0023	16384	38	-	34.56	36	0.15	34.56	9	0.19
Gaussian + Packet Loss						0.3	34.53	1161	4.94	34.53	235	4.98
						11	28.91	5842	24.84	28.91	1170	24.8
Packet Loss only	8×8	23	0.0022	4096	9	-	35.47	26	0.11	35.47	5	0.11
Gaussian + Packet Loss						0.3	35.46	1172	4.98	35.46	231	4.9
						11	29.23	5931	25.21	29.23	1203	25.5
Packet Loss only	16×16	16	0.0019	1024	2	-	34.95	25	0.11	34.95	5	0.11
Gaussian + Packet Loss						0.3	34.91	1148	4.88	34.91	229	4.85
						11	28.88	5932	25.22	28.88	1190	25.23

error rates; it can be seen that the determining factor in all cases is the magnitude (standard deviation) of Gaussian noise.

Detection performance

We trained the neural network that classifies distorted stego-free vs. distorted stego images for all three steganographic algorithms and all noise parameters considered. Table 2 reports the number of epochs and the detection accuracy (relative frequency of correct classification) of this network on the unseen prediction set. The training took roughly 4 minutes per epoch on a Tesla K80 GPU with 12 GB memory. The results have been obtained using early stopping. As an example, training without early stopping as in [6] would take 108 epochs (or 7.2 hours) for WOW with $\alpha = 0.1$ and a combination of Gaussian noise and packet loss, and the network would achieve 72.95% accuracy. The results in Table 2 (with early stopping) are: 74.85%, obtained in 94 epochs or 6.3 hours.

From Table 2, one can see that the employed NN detects steganography insertion very well for high values of parameter α ($0 < \alpha < 1$), which determines how many bits of an image are used for steganographic information. Fig. 7 shows training

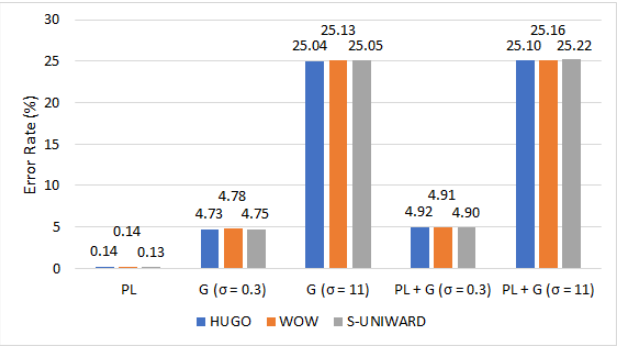
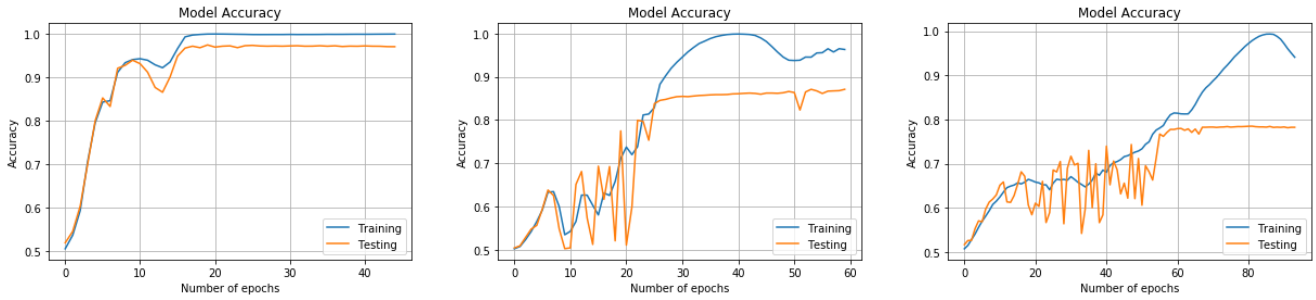


Figure 6. *Average error rates for different stego algorithms and noise parameters (G = Gaussian, PL = packet loss)*

Table 2: Detection accuracy for different scenarios

	Stego algorithm	Payload param. α	Number of epochs	Detection Accuracy [%]
Gaussian noise only	HUGO	0.1	84	74.05
		0.4	60	97.5
	WOW	0.1	84	75.58
		0.4	58	96.55
	S-UNIWARD	0.1	75	74.98
		0.4	60	95.75
Packet loss only	HUGO	0.1	95	77.93
		0.4	60	96.45
	WOW	0.1	85	78.63
		0.4	50	93.48
	S-UNIWARD	0.1	80	76.9
		0.4	58	95.6
Gaussian noise and packet loss	WOW	0.1	94	74.85
		0.2	60	81.05
		0.3	54	86.6
		0.4	45	94.1
	HUGO	0.1	98	73.2
		0.4	47	96.68
	S-UNIWARD	0.1	97	73.65
		0.4	48	94.45

and test accuracy for one algorithm "WOW" and a combination of Gaussian noise and packet loss (the other algorithms behaved similarly). Furthermore, from Table 2, it can be seen that irrespective of the stego algorithm, the combination of both Gaussian noise and packet loss has the greatest impact on detection accuracy followed by Gaussian noise only and packet loss only.



(a) (b) (c)

Figure 7. *Training accuracy (classification of the training set) and test accuracy (classification of the test set) of WOW stego algorithm with parameter α set to (a) 0.4 (resulting in 23,754 stego bits per 512×512-pixel image on average); (b) 0.2 (10,379 stego bits per image); (c) 0.1 (4,643 stego bits per image).*

Conclusions and Future Work

We have considered the problem of steganalysis over noisy channels, pointing out the implications of noise to both: performing steganography and detecting it. We improved the deep learning strategy and observed good detection accuracy on realistic images. Our results indicate that steganalysis becomes feasible when the amount of stego information in an image increases. At the same time, even moderate noise with limited effect on the quality of the images themselves can have grave consequences for the steganographic channel, leading to very large error rates. While an adversary could find a fault-tolerant scheme to still transmit the information, it would require a large number of redundant bits and therefore be detectable by our deep-learning approach.

In the future, we plan to design countermeasures against malicious steganography based on strategically inducing errors on the communication channel while avoiding too harsh consequences for the image quality. We would also like to understand better what an adversary has to do to overcome the desynchronization problem, and prevent him from doing so.

References

[1] T. Morkel, J. H. P. Eloff, and M. S. Olivier. An overview of image steganography. In *ISSA*, pages 1–11. ISSA, Pretoria, South Africa, 2005.

[2] B. Dolhansky, R. Howes, B. Pflaum, N. Baram, and C. Canton-Ferrer. The deepfake detection challenge (DFDC) preview dataset. *CoRR*, abs/1910.08854, 2019.

[3] T. Filler and J. J. Fridrich. Gibbs construction in steganography. *IEEE Trans. Information Forensics and Security*, 5(4):705–720, 2010.

[4] V. Holub and J. J. Fridrich. Designing steganographic distortion using directional filters. In *WIFS*, pages 234–239. IEEE, 2012.

[5] V. Holub, J. J. Fridrich, and T. Denemark. Universal distortion function for steganography in an arbitrary domain. *EURASIP J. Information Security*, 2014:1, 2014.

[6] M. Salomon, R. Couturier, C. Guyeux, J.-F. Couchot, and J.M. Bahi. Steganalysis via a convolutional neural network using large convolution filters for embedding process with same stego key: A deep learning approach for telemedicine. *European Research in Telemedicine*, 6(2):79–92, 2017.

[7] Y. Qian, J. Dong, W. Wang, and T. Tan. Deep learning for steganalysis via convolutional neural networks. In *Media Watermarking, Security, and Forensics*, volume 9409 of *SPIE Proceedings*, page 94090J. SPIE, 2015.

[8] L. Pibre, J. Pasquet, D. Ienco, and M. Chaumont. Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover sourcemismatch. In *Media Watermarking, Security, and Forensics*, pages 1–11. Ingenta, 2016.

[9] S. Baluja. Hiding images in plain sight: Deep steganography. In *NIPS*, pages 2069–2079, 2017.

[10] D. Ye, S. Jiang, S. Li, and C. Liu. Faster and transferable deep learning steganalysis on GPU. *J. Real-Time Image Processing*, 16(3):623–633, 2019.

[11] T. Pevný, T. Filler, and P. Bas. Using high-dimensional image models to perform highly undetectable steganography. In *Information Hiding*, volume 6387 of *LNCS*, pages 161–177. Springer, 2010.

[12] B. Kaur, A. Kaur, and J. Singh. Steganographic approach for hiding image in dct domain. *International Journal of Advances in Engineering & Technology*, 1:72–78, 2011.

[13] Y. Cao, X. Zhao, D. Feng, and R. Sheng. Video steganography with perturbed motion estimation. In *Information Hiding*, volume 6958 of *LNCS*, pages 193–207. Springer, 2011.

[14] M. Hussain, A. W. A. Wahab, Y. I. B. Idris, A. T. S. Ho, and K.-H. Jung. Image steganography in spatial domain: A survey. *Sig. Proc.: Image Comm.*, 65:46–66, 2018.

[15] I. Koren and C. Mani Krishna. *Fault-tolerant systems*. Morgan Kaufmann, 2010.

[16] V. I. Korzhik, G. Morales-Luna, and K. Loban. Stegosystems based on noisy channels. *IJCSA*, 8(1):1–13, 2011.

[17] V. Korzhik, G. Morales-Luna, K. Loban, and I. Marakova-Begoc. Undetectable spread-time stegosystem based on noisy channels. In *Proc. Int'l Multiconf. Computer Science and Information Technology*, pages 723–728. IEEE, 2010.

[18] P. Bas, T. Filler, and T. Pevný. "Break our steganographic system": The ins and outs of organizing BOSS. In *Information Hiding*, volume 6958 of *LNCS*, pages 59–70. Springer, 2011.

[19] D.-T. Dang-Nguyen, C. Pasquini, V. Conotter, and G. Boato. RAISE: a raw images dataset for digital image forensics. In *MMSys*, pages 219–224. ACM, 2015.

[20] Steganographic algorithms. http://dde.binghamton.edu/download/, (Accessed December 24, 2019).

[21] O. Hadar, R. Shmueli, R. Huber, and M. Huber. Effects of compression parameters on the perceived quality of video stream over a lossy internet protocol network. *Optical Engineering*, 45(8):087003, 2006.

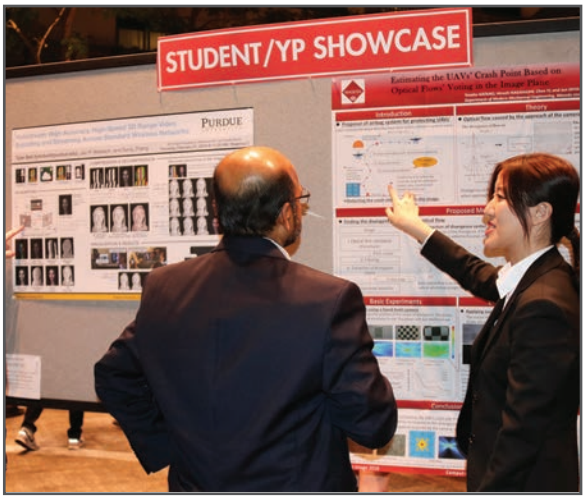
JOIN US AT THE NEXT EI!

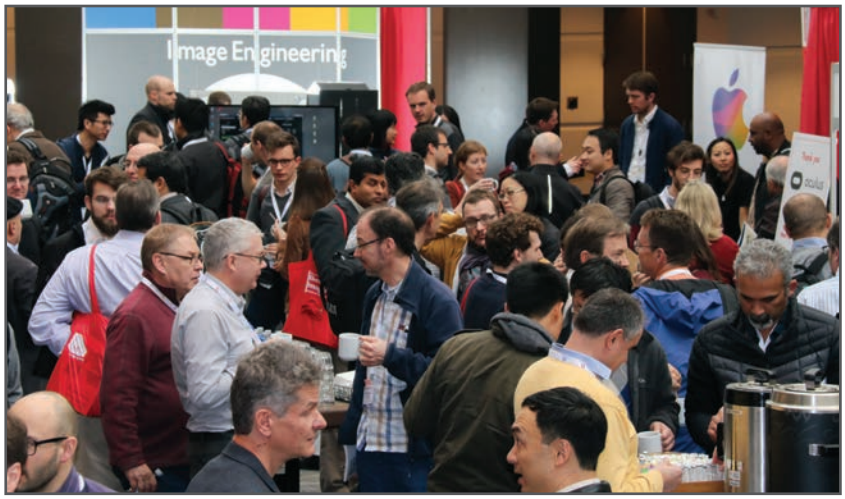
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org