https://doi.org/10.2352/ISSN.2470-1173.2018.07.MWSF-121

Steganalyzing Images of Arbitrary Size with CNNs

Clement Fuji Tsang and Jessica Fridrich, Department of ECE, SUNY Binghamton, NY, USA, clement.fuji_tsang@utt.fr, fridrich@binghamton.edu

Abstract

Convolutional neural networks offer much more accurate detection of steganography than the outgoing paradigm - classifiers trained on rich representations of images. While training a CNN is scalable with respect to the size of the training set, one cannot directly train on images that are too large due to the memory limitations of current GPUs. Most leading network architectures for steganalysis today require the input image to be a small tile with 256×256 *or* 512×512 *pixels. Because detecting the presence of steganographic embedding changes really means detecting a very weak noise signal added to the cover image, resizing an image before presenting it to a CNN would be highly suboptimal. Applying the tile detector on disjoint segments of a larger image and fusing the results bring a plethora of new problems of how to properly fuse the outputs. In this paper, we propose a different solution to this problem based on modifying an existing leading network architecture for steganalysis in the spatial domain, the YeNet, to output statistical moments of feature maps to the fully-connected classifier part of the network. On experiments in which we adjust the payload with image size according the square root law for constant statistical detectability, we demonstrate that the proposed architecture can be trained to steganalyze images of various sizes without any or only a small loss with respect to detectors trained for a fixed image size.*

Introduction

Steganography allows the sender to communicate secretly and covertly over insecure channels by hiding the message in an innocuous looking cover object. The goal of steganalysis is to detect the presence of secretly embedded messages. Recently, steganalysis detectors implemented as deep Convolutional Neural Networks (CNNs) have appeared [20, 1, 28, 27, 3, 26, 30] and quickly established themselves as superior to the previous detection paradigm, detectors built as binary classifiers [17, 5, 18] trained on examples of cover and stego images represented with high-dimensional (rich) features [7, 16, 22]. The most notable contributions in this direction include the pioneering architecture proposed by Qian [20, 1], the XuNet [28, 27], JPEG-phase-aware XuNet [3], deep network targeted to J-UNIWARD [26], and the breakthrough YeNet [30], with a significantly better performance for spatial-domain steganography in comparison to conventional detectors.

While the training of CNN detectors is far more scalable with respect to the size of the training set, because of the limited memory of GPUs on which CNNs are trained, it is not possible to train such detectors on images that are too large. Current technology allows computationally feasible training on relatively small tiles, typically 256×256 or 512×512 pixels. In particular, images with several megapixels would not fit on the GPU memory even when using "mini-batches" consisting of a single image. Small mini-batches would also compromise the stability of stochastic gradient descent algorithms because of noisy gradients, which would negatively affect the network convergence and its performance. This brings a natural question, which is how should large images be steganalyzed with a detector that can only accept a small tile on its input.

In pattern recognition applications, this problem has typically been approached by resizing the input image directly to the desired size and training with data augmentation to be robust w.r.t the size of the pattern. Another possibility is to use an approach that is already invariant to scale [29]. However, this would not be a good practice in steganalysis as the patterns (the steganographic embedding changes) are very weak, independent, and, by definition, pixel-wise. The classification is thus a quantitative analysis of such subtle noise-like patterns. Downsizing before classification would compromise the detector accuracy because it would "average out" adjacent embedding changes and thus decrease the signal to noise ratio between the cover image and the steganographic signal.

In principle, one could apply the tile detector on disjoint (or overlapping) tiles covering the analyzed image and fuse/pool the results, which is the problem of pooled steganalysis introduced in [9, 10, 12, 15, 13] and studied in [19, 4, 21]. Finding a proper pooling function, however, would have to take into consideration the dependencies among tiles, which appears to be a hard problem further complicated by the fact that the distribution of CNN output test statistic is highly non-Gaussian. The pooling seems like a task that, too, should best be outsourced to machine learning.

In this paper, we investigate an alternative option by selecting an architecture for the network that is already scalable w.r.t. the size of the analyzed image. This can be achieved by feeding *statistical moments* of the feature maps outputted by the last layer before the fully connected (classifier) part of the network also called the IP layer (inner product). The belief is that, due to self-similarity of natural images [23, 24, 25], the front part of the network will be a "universal feature extractor" and one only needs to retrain the IP layer of the network to adapt to a different input image size. Indeed, non-linear moments, such as maximum, minimum, and variance of feature maps, will indirectly inform the IP layer about the input image size

and its resolution. For example, variance will be smaller for high-resolution input images while the order moments will increase with increased size of the input image. Additionally, one could also supply the number of pixels as an additional input to the IP layer.

We evaluate the effectiveness of the proposed approach in two different ways – by direct comparison on image sizes for which it is feasible and possible to directly train the CNN and by indirect comparison while leveraging the square root law [14, 6, 11]. Note that, depending on the network structure, it is not possible to arbitrarily decrease the size of the input tile without modifying the network architecture by removing layers. For example, as of January 2018, the most successful CNN architecture for detection spatial domain steganography is the YeNet [30], which outputs 16 3×3 feature maps before the IP layer. The smallest input image is thus 128×128. The largest input image on which it is computationally feasible to train the YeNet with a single GPU with 12 GB memory[1] is 512×512. Expanding to larger sizes could be done by decreasing the embedding change rate for larger images to keep the statistical detectability constant according to the square root law.

In the next section, we describe the network used in our experiments, the YeNet, and its modifications to make it more amenable for steganalysis in images of various size. In particular, the IP layer is fed with moments of the 7×7 output feature maps. In Section "Experiments", we report the results of two types of experiments aimed at assessing the performance of the CNN adapted to detect steganography in images of a different size. This section also contains all details regarding the CNN training, the image datasets, and other relevant experimental setup for our experiments. The last section summarizes the paper.

Network architecture

We used a modified YeNet [30] without the selection-channel-aware part shown in Figure 1. One of the modifications is the addition of a batch normalization layer after each ReLU. We also reduced the stride in the last 9th convolutional layer before classification to one, which made the size of the 16 features before the IP layer to be 7×7 rather than 3×3 as in the original YeNet. Furthermore, we made the moments of these feature maps, their sample average, sample variance, maximum, and minimum, as the outputs of this layer. According to our experience, adding the batch normalization after each ReLU prevents the network from over-fitting and gives a slightly better detection accuracy.

Moments extraction and training

The maximum, minimum, variance, and average moments were computed from each of the last 16 feature maps. The variance was considered as a constant and its derivative was ignored during backpropagation. All $4 \times 16 = 64$ moments were then fed directly to the IP layer (Figure 1).

[1]The GPU memory size for Titan X and Xp, Tesla K40 and K80, and GTX 1080 Ti (11 GB).

The combination of variance and the order moments (minimum and maximum) inform the IP layer about the resolution and size of the input image, respectively. Here, we distinguish between the image size, the number of pixels, and the image resolution, which relates to the native resolution of the image at acquisition. For example, a small crop from a high resolution image will keep the smooth nature of a high resolution image (smaller variance of feature maps) despite being small. On the other hand, a resized image (with antialiasing turned off) will have a significantly increased level of detail, which will translate into a larger variance. The order moments monotonically decrease with cropping. Consequently, we believe the four moments provide enough information to the IP layer to allow the CNN adjust itself to accurately steganalyze images of arbitrary size and resolution.

In this paper, we will be using two versions of the modified YeNet – one with a single-layer IP and one with a two-layer IP. The single-layer IP will be used for training a "feature extractor" or "tile detector" while the two-layer IP version will be used to make the network handle images of different sizes because changing the size may change the classification based on extracted moments to a non-linear problem. The final modified YeNet network architecture is shown in Figure 1.

Experiments

In the first subsection, we propose an approach for adapting a network trained on small tiles to handle larger images of a fixed size with an accuracy approximately equal to a network directly trained on larger images, if such network could be built. This was achieved by first training the tile detector on the smallest size of 256×256, cropped from larger original sizes. Cropping was used instead of resizing because we intend to adjust the payload for larger image sizes based on the square root law to preserve the statistical detectability across various image sizes and thus allow us to assess the performance of networks on large images. Then, to obtain a detector for a fixed larger size, we retrained only the IP layers of this network with the same "feature extractor," the front part of the network before the IP layers. This retraining was achieved in two phases – first the moments were extracted from the larger images and then a two-layer IP was trained on mini-batches of moments (moments from 1000 images in each mini-batch). We call this detector TRIP because of the fixed **T**ile detector front part and the **R**etrained **IP** layer.

Since the payload/change rate is adjusted for a constant statistical detectability across different sizes, we also added the results obtained when retraining the tile detector for the smaller payload inserted in the larger images and then retraining just the IP layers on moments extracted from larger images. We call it RTRIP (**R**etrained **T**ile detector and **R**etrained **IP**).

Our main goal is to show how a network trained on one size can be adapted to detect on larger sizes without any loss on detection accuracy. The comparison between the TRIP and RTRIP detectors will also inform us about how universal the feature extractor trained on small tiles

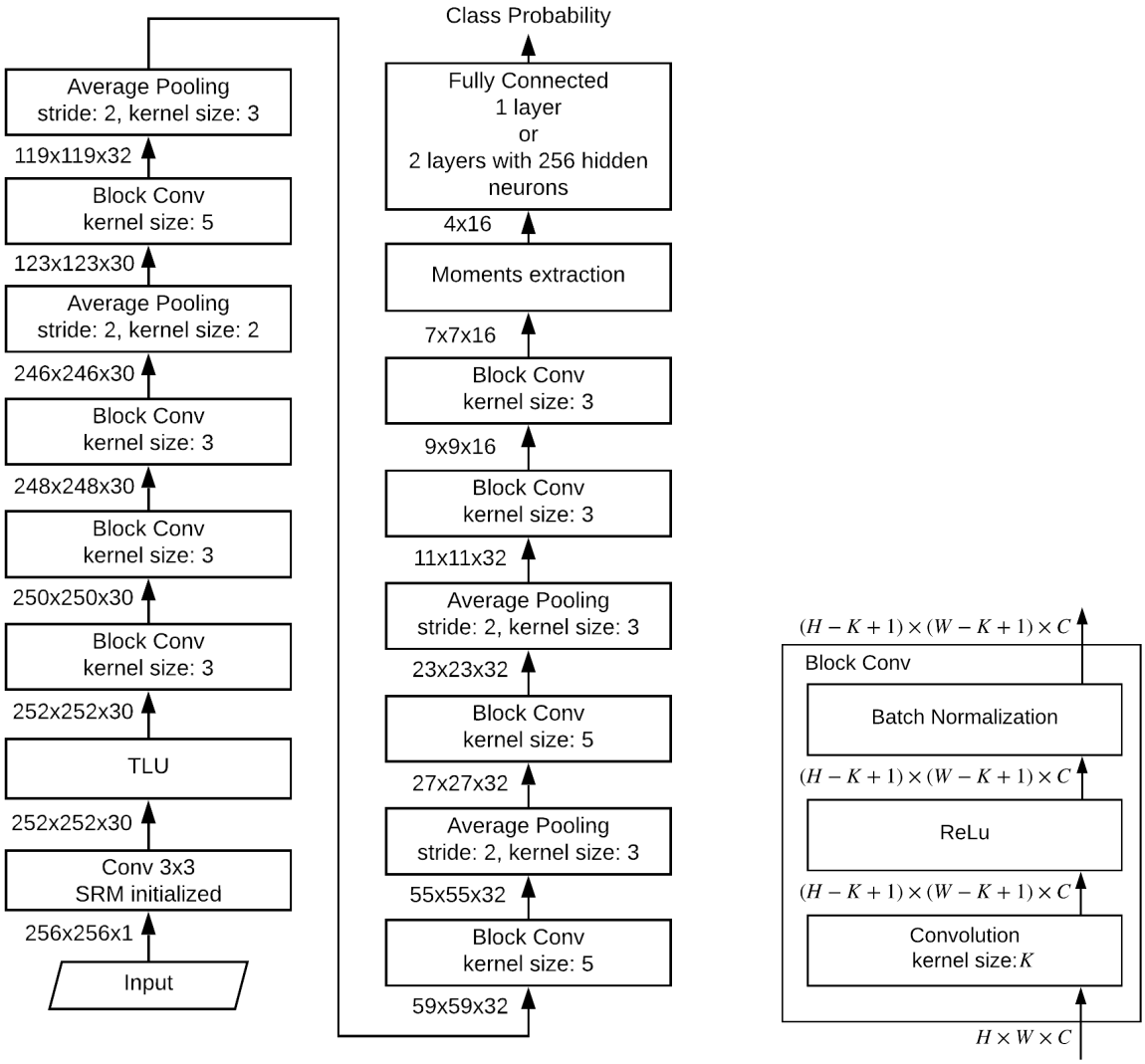

Figure 1. *Left: Modified YeNet used in our work. Right: architecture of each 'Block Conv' layer.*

is across images of various sizes and whether its retraining for a different payload is needed.

In the second subsection, we propose and study an approach for building a truly Size-Independent Detector (SID) that can accept *arbitrarily sized* input images and return as accurate an answer as TRIP or RTRIP detectors built for a *fixed* input image size.

Detecting in large images of a fixed size

The datasets for the first batch of experiments in this section were prepared from the original 10,000 BOSSbase 1.01 [2] native resolution images resized to 1024×1024 using the `convert` tool from ImageMagick, and then split into 6400 / 1600 / 2000 images, respectively, for training / validation / testing. We then generated from each image *in the training set* smaller images by cropping each image into four 512×512 images and 16 256×256 images. For validation and testing, only one central crop and one "smart crop" was obtained from each large image so that its histogram of local variance (computed from 3×3 blocks) was the most similar (in L1 norm) to the histogram of the local variance of the entire 1024×1024 image. This was done to obtain sources that are more similar to the original source of 1024×1024 images because cropping may change the source properties and produce images with singular content, such as completely saturated images or images of just blue sky. Thus, the final number of training / validation / testing images for 256×256 and 512×512 images were 102,400 / 1600 / 2000 and 25,600 / 1600 / 2000, respectively.

Experiments were carried out on the steganographic algorithm WOW [8] and the non-adaptive Least Significant Bit Matching (LSBM). The LSBM was added to the experiments because it is easier to apply the square root law for a non-adaptive embedding scheme. For constant statistical detectability, the payload for WOW was adjusted so that the change rate averaged over the whole test dataset followed the required scaling dictated by the square root law. For example, the average change rate for 512×512 images was twice smaller than for 256×256 images. For the non-adaptive LSBM, the change rate was fixed for each image size across all stego images from the database. We note that for optimally coded LSBM the relationship between the change rate β and relative payload size α in bits per pixel (bpp) is defined by the rate-distortion bound for non-adaptive ternary embedding: $\alpha = h_3(\beta)$, where

$$h_3(x) = -x \log_2(x/2) - (1-x) \log_2(1-x) \tag{1}$$

is the ternary entropy function when β is the total change rate, the relative number of changes by either $+1$ or -1 per pixel. Figure 1 summarizes the payloads and change rates used for WOW and LSBM.

Size	256×256	512×512	1024×1024
LSBM	0.28 / 0.04	0.16 / 0.02	0.09 / 0.01
WOW	0.4 / 0.082	0.22 / 0.041	0.12 / 0.021

Table 1. Payload for WOW (bpp) / change rate for LSBM (change per pixel) for different image sizes used in experiments with RTRIP and TRIP detectors.

For the RTRIP detector, the modified YeNet was trained on 256×256 crops at the largest payload 0.4 bpp for WOW and 0.04 change rate for LSBM (see Table 1) using Adadelta [31] with learning rate set to 0.4, which was then decayed to 0.08 once the accuracy plateaued. We applied curriculum learning to fine-tune this tile detector (the "feature extractor") to the payload embedded in larger images followed by extracting moments (eight sets of four moments were extracted for each image rotated by 90 degrees and mirrored) by feeding this network with 512×512 or 1024×1024 stego and cover images and finally training just the two-layer IP to obtain the RTRIP detector for larger image sizes. For the TRIP detector, only the two-layer IP part was retrained on moments extracted from the front part of the network (the tile detector). For training, the embedding was done on-the-fly. It was fixed for validation and testing. The training included augmentation (random application of rotation and mirroring) and shuffling of the training database after each epoch.

The detection performance was evaluated using the minimal total error probability under equal priors

$$P_{\mathrm{E}} = \min_{P_{\mathrm{FA}}} \frac{1}{2}(P_{\mathrm{FA}} + P_{\mathrm{MD}}) \tag{2}$$

on the testing set, where P_{FA} and P_{MD} stand for the false-alarm and missed detection probabilities.

The results shown in Table 2 indicate the effectiveness of the RTRIP and TRIP detectors, which perform as well as they can across a range of image sizes (65,000 pixels to one megapixel). For both steganographic algorithms and all sizes, the RTRIP detector performed as well as the TRIP detector, confirming thus the universality of the "feature extractor" from the tile detector. For LSBM, the scaling according to the square root law indeed leads to an approximately constant statistical detectability. Though, a wider range of scales needs to be tested for a more conclusive evidence (see the next section). For WOW, the detection errors for larger sizes seem to lack behind what one would expect according to the square root law. Note, however, that this is true even for detectors trained directly on larger sizes. This indicates a potential flaw in the adjustment of payloads for WOW as averages over the source. In fact, the scaling of the payload for content-adaptive algorithms should be done on a per-image basis, depending on the type of the warden (detector).[2]

To verify that the proposed TRIP and RTRIP detectors scale well on a wider range of image sizes, we now report the results for images obtained by resizing the native resolution BOSSbase 1.01 images using the same `convert` script to 2000×2000. This time, we only evaluate the performance of the tile detector trained on 64 256×256 crops from the large images when validating/testing on 1600/2000 smart crops prepared as explained above. Table 4 shows the results for the TRIP and RTRIP detectors on both embedding algorithms when adjusting the payload as described in Table 3. For LSBM, the RTRIP detector scales to the larger images without any loss on detection.

[2] A. Ker, personal communication, 2017.

LSBM	central 256×256	central 512×512	smart 256×256	smart 512×512	1024×1024
Direct training	0.1487	0.1332	0.1197	0.1092	X
RTRIP	0.1472	0.1330	0.1177	0.1068	0.0940
TRIP	0.1472	0.1260	0.1177	0.1100	0.1120

WOW	central 256×256	central 512×512	smart 256×256	smart 512×512	1024×1024
Direct training	0.1273	0.1547	0.1197	0.1455	X
RTRIP	0.1385	0.1535	0.1168	0.1310	0.1445
TRIP	0.1385	0.1530	0.1168	0.1303	0.1583

Table 2. Detection error P_{E} for RTRIP and TRIP detectors. Direct training means that the detector was trained on full size images. The RTRIP detectors were obtained by retraining the tile detector via curriculum training for the payload residing in the larger images while the TRIP detector uses the unchanged feature extractor from the tile detector while only retraining its IP part. The symbol 'X' marks practically unachievable results due to limited GPU memory.

The TRIP detector, however, performs worse. This is likely due to the rather large difference in the change rate, and thus the "density of changes," between the two image sizes. In other words, when the feature extractor was retrained on the smaller change rate via curriculum training, no loss of performance on larger images was observed. In contrast, for WOW both the TRIP and RTRIP detectors perform equally on large images. We believe this is because of the content-adaptivity of WOW, which keeps the density of changes concentrated in textured / noisy regions while avoiding smooth regions. In other words, for WOW the selection channel looks more similar across varying payloads than for LSBM. Finally, for WOW the discrepancy between the detection error for the tiles and the large images is likely due to improperly adjusting the payload for constant statistical detectability.

Size	256×256	2000×2000
LSBM	0.28 / 0.04	0.051 / 0.00512
WOW	0.4 / 0.074	0.066 / 0.0095

Table 3. Payload (bpp) for WOW / change rate for LSBM (change per pixel) for different images sizes adjusted for constant statistical detectability according to the square root law for the TRIP and RTRIP detectors when cropping 256×256 images from BOSSbase 1.01 resized to 2000×2000.

Steganalyzing images of varying size

We now move towards building a truly size independent detector (SID) that can steganalyze images of arbitrary size. The experiments here were conducted on BOSSbase resized to 1024×1024 as in the first experiment in the previous section. First, we trained the modified YeNet on 256×256 crops from the 1024×1024 images (16 256×256 tiles per one large image). The stego images were all embedded with the same payload of 0.12 bpp for WOW and modified with the same change rate 0.01 for LSBM. Then, the front part of the network, the feature extractor, was fixed and the two-layer IP part was retrained on moments extracted from images from the training set prepared by random cropping executed in the following manner. For each 1024×1024 image in the training dataset, we obtained 32 random square crops of random size between 256×256 and 1024×1024. First a random integer $r\in[256,1024]$ was generated and then the coordinates of the upper left corner of the crop (x,y), $x,y\in[1,1024-r]$ were also uniformly randomly generated to make sure the cropped image fits the host 1024×1024 image. This SID was then tested on 256×256 and 512×512 smart crops, and 1024×1024 original images previously made from the validation / test dataset. We also tested whether adding the number of pixels N_p (scaled by 100,000) as an additional input to the IP layers can improve the detection accuracy.

The results shown in Table 5 indicate that the SID performs as accurately as the RTRIP detector trained for a fixed image size. Note that adding the number of pixels N_p as an additional input to the IP layer does not improve the detection. This indicates that the features/moments already contain information about the input image size. To clarify the experiments, we summarize that the rows for 'RTRIP, fixed size' mean that the feature extractor (tile detector) was trained on 256×256 images and then, for the larger sizes just a two-layer IP was retrained on moments extracted from 512×512 and 1024×1024 images, establishing thus a benchmark for the SID. The SIDs were trained only once on a mixture of image sizes as explained above and tested on smart crops of fixed size to allow a comparison with the detector trained specifically for this size, the row 'RTRIP, fixed size' in the table.

Conclusions

Steganalysis using CNNs clearly outperforms detectors built using the outgoing detection paradigm – classifiers trained on rich media features. CNN detectors enjoy many other important advantages, such as scalability w.r.t. the training set size and better detection rates for low false alarm rates. However, due to the nature of the detector and the limits of current hardware, it is not possible to directly train on large images. The leading CNN detectors for steganalysis need a rather small tile on their input. This poses a natural question as to how one should steganalyze a large image.

While it is possible to resize the input image or apply the tile detector on disjoint or overlapping tiles and fuse the results, such approaches will inevitably be suboptimal due to suppressing the noise-like stego signal when downsampling or when fusing outputs with complicated mutual

LSBM	smart 256×256	2000×2000
Direct training	0.104	X
TRIP	0.105	0.150
RTRIP	0.105	0.098

WOW	smart 256×256	2000×2000
Direct training	0.094	X
TRIP	0.093	0.172
RTRIP	0.093	0.185

Table 4. Detection error P_{E} for RTRIP and TRIP detectors when adjusting the tile detector for detection in 2000×2000 images. The payload embedded in the images is listed in Table 3. The symbol 'X' marks practically unachievable results due to limited GPU memory.

		Tested on fixed size		
Algorithm	Detector type, training	256×256	512×512	1024×1024
LSBM	RTRIP, fixed size	0.253	0.156	0.0940
	SID, random size	0.243	0.163	0.0856
	SID+N_p, random size	0.244	0.160	0.0837
WOW	RTRIP, fixed size	0.261	0.204	0.1445
	SID, random size	0.259	0.197	0.1390
	SID+N_p, random size	0.253	0.197	0.1380

Table 5. Detection error P_{E} for the RTRIP detector trained on images of one size with IP retrained for a larger fixed size and SID trained on images of random sizes without and with the number of pixels N_p as the input to the IP layer. All detectors are always tested on a fixed size. Payload 0.12 bpp for WOW and change rate 0.01 for LSBM.

dependencies. In this paper, we propose a different approach in which we first modify an existing CNN detector to output statistical moments of feature maps that enter the fully-connected IP layers of the network (the average, minimum, maximum, and variance). To build a network capable of detecting steganography in large images, the network with moments is first trained on small tiles. The front part of this detector, which is considered as a "feature extractor" or, better, "moment extractor," is used to extract moments from the larger images. Then, a two-layer IP layer is trained on these moments to obtain a detector for large images. We conducted experiments with detectors built for a fixed large size as well as detectors meant to detect steganography in images of a wider range of sizes. To assess their performance, we adjusted the payload in large images for constant statistical detectability according to the square root law. We demonstrated that this approach to building a size independent detector is effective as well as accurate.

All code used to produce the results in this paper, including the network configuration files are available from `http://dde.binghamton.edu/download/`.

Acknowledgments

The work on this paper was supported by the Air Force Office of Scientific Research under the research grant FA9950-12-1-0124. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of AFOSR or the U.S. Government.

References

[1] Learning and transferring representations for image steganalysis using convolutional neural network. In *IEEE International Conference on Image Processing (ICIP)*, pages 2752–2756, September 25-28, 2016.

[2] P. Bas, T. Filler, and T. Pevný. Break our steganographic system – the ins and outs of organizing BOSS. In T. Filler, T. Pevný, A. Ker, and S. Craver, editors, *Information Hiding, 13th International Conference*, volume 6958 of Lecture Notes in Computer Science, pages 59–70, Prague, Czech Republic, May 18–20, 2011. Springer Berlin Heidelberg.

[3] M. Chen, V. Sedighi, M. Boroumand, and J. Fridrich. JPEG-phase-aware convolutional neural network for steganalysis of JPEG images. 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.

[4] R. Cogranne. A sequential method for online steganalysis. In *Information Forensics and Security (WIFS), IEEE 7th International Workshop on*, Rome, Italy, November 2015.

[5] R. Cogranne and J. Fridrich. Modeling and extending the ensemble classifier for steganalysis of digital images using hypothesis testing theory. *IEEE Transactions on Information Forensics and Security*, 10(2):2627–2642, December 2015.

[6] T. Filler, A. D. Ker, and J. Fridrich. The Square Root Law of steganographic capacity for Markov covers. In N. D. Memon, E. J. Delp, P. W. Wong, and

J. Dittmann, editors, *Proceedings SPIE, Electronic Imaging, Media Forensics and Security*, volume 7254, pages 08 1–11, San Jose, CA, January 18–21, 2009.

[7] 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.

[8] V. Holub and J. Fridrich. Designing steganographic distortion using directional filters. In *Fourth IEEE International Workshop on Information Forensics and Security*, Tenerife, Spain, December 2–5, 2012.

[9] A. D. Ker. Batch steganography and pooled steganalysis. In J. L. Camenisch, C. S. Collberg, N. F. Johnson, and P. Sallee, editors, *Information Hiding, 8th International Workshop*, volume 4437 of Lecture Notes in Computer Science, pages 265–281, Alexandria, VA, July 10–12, 2006. Springer-Verlag, New York.

[10] A. D. Ker. Batch steganography and the threshold game. In E. J. Delp and P. W. Wong, editors, *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX*, volume 6505, pages 04 1–13, San Jose, CA, January 29–February 1, 2007.

[11] A. D. Ker. A capacity result for batch steganography. *IEEE Signal Processing Letters*, 14(8):525–528, 2007.

[12] A. D. Ker. Perturbation hiding and the batch steganography problem. In K. Solanki, K. Sullivan, and U. Madhow, editors, *Information Hiding, 10th International Workshop*, volume 5284 of Lecture Notes in Computer Science, pages 45–59, Santa Barbara, CA, June 19–21, 2008. Springer-Verlag, New York.

[13] A. D. Ker. Locally square distortion and batch steganographic capacity. *International Journal of Digital Crime and Forensics*, 1(1):29–44, 2009.

[14] A. D. Ker, T. Pevný, J. Kodovský, and J. Fridrich. The Square Root Law of steganographic capacity. In A. D. Ker, J. Dittmann, and J. Fridrich, editors, *Proceedings of the 10th ACM Multimedia & Security Workshop*, pages 107–116, Oxford, UK, September 22–23, 2008.

[15] A. D. Ker and Tomas Pevný. Batch steganography in the real world. In J. Dittmann, S. Craver, and S. Katzenbeisser, editors, *Proceedings of the 14th ACM Multimedia & Security Workshop*, pages 1–10, Coventry, UK, September 6–7, 2012.

[16] 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.

[17] J. Kodovský, J. Fridrich, and V. Holub. Ensemble classifiers for steganalysis of digital media. *IEEE Transactions on Information Forensics and Security*, 7(2):432–444, 2012.

[18] I. Lubenko and A. D. Ker. Steganalysis with mismatched covers: Do simple classifiers help. In J. Dittmann, S. Katzenbeisser, and S. Craver, editors, *Proc. 13th ACM Workshop on Multimedia and Security*, pages 11–18, Coventry, UK, September 6–7 2012.

[19] T. Pevný and I. Nikolaev. Optimizing pooling function for pooled steganalysis. In *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*, November 2015.

[20] 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, pages 0J 1–10, San Francisco, CA, February 8–12, 2015.

[21] V. Sedighi, R. Cogranne, and J. Fridrich. Practical strategies for content-adaptive batch steganography and pooled steganalysis. In *Proc. IEEE ICASSP*, New Orleans, LA, March 5–9, 2017.

[22] X. Song, F. Liu, C. Yang, X. Luo, and Y. Zhang. Steganalysis of adaptive JPEG steganography using 2D Gabor filters. In A. Alattar, J. Fridrich, N. Smith, and P. Comesana Alfaro, editors, *The 3rd ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '15, Portland, OR, June 17–19, 2015.

[23] A. Turiel, G. Mato, and N. Parga. The self-similarity properties of natural images resemble those of turbulent flows. *Physical Review Letters*, 80(5):1098–1101, 1998.

[24] A. Turiel and N. Parga. The multi-fractal structure of contrast changes in natural images: From sharp edges to textures. *Neural Computation*, 12:763–793, 2000.

[25] A. Turiel, N. Parga, D.L. Ruderman, and T.W. Cronin. Multiscaling and information content of natural color images. *Physical Review E*, 62(1):1138–1148, 2000.

[26] 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.

[27] G. Xu, H.-Z. Wu, and Y. Q. Shi. Ensemble of CNNs for steganalysis: An empirical study. In F. Perez-Gonzales, F. Cayre, and P. Bas, editors, *The 4th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '16, pages 5–10, Vigo, Spain, June 20–22, 2016.

[28] G. Xu, H. Z. Wu, and Y. Q. Shi. Structural design of convolutional neural networks for steganalysis. *IEEE Signal Processing Letters*, 23(5):708–712, May 2016.

[29] Y. Xu, T. Xiao, J. Zhang, K. Yang, and Z. Zhang. Scale-invariant convolutional neural networks. *CoRR*, abs/1411.6369, 2014.

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

[31] M. D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

Author Biography

Clement Fuji Tsang is a MSc degree student in Electrical and Computer Engineering at the University of Technology of Troyes, in France, expected to be graduating in

July 2018. His research interest are in the field of deep learning applied to machine perception and reasoning and hardware / low-level optimization applied to machine learning algorithms.

Jessica Fridrich is Distinguished Professor of Electrical and Computer Engineering at Binghamton University. She received her PhD in Systems Science from Binghamton University in 1995 and MS in Applied Mathematics from Czech Technical University in Prague in 1987. Her main interests are in steganography, steganalysis, and digital image forensics. Since 1995, she has received 20 research grants totaling over $11 mil that lead to more than 180 papers and 7 US patents.