Ensemble of CNNs for Steganalysis: An Empirical Study

Guanshuo Xu
New Jersey Institute of Technology
University Heights
Newark, NJ, USA 07102
+1-862-371-4304
gx3@njit.edu

Han-Zhou Wu
Southwest Jiaotong University
High-tech Zone, West Park
Chengdu 611756, P.R. China
+86-159-2811-3628
h.wu.phd@ieee.org

Yun Q. Shi
New Jersey Institute of Technology
University Heights
Newark, NJ, USA 07102
+1-973-596-3501
shi@njit.edu

ABSTRACT

There has been growing interest in using convolutional neural networks (CNNs) in the fields of image forensics and steganalysis, and some promising results have been reported recently. These works mainly focus on the architectural design of CNNs, usually, a single CNN model is trained and then tested in experiments. It is known that, neural networks, including CNNs, are suitable to form ensembles. From this perspective, in this paper, we employ CNNs as base learners and test several different ensemble strategies. In our study, at first, a recently proposed CNN architecture is adopted to build a group of CNNs, each of them is trained on a random subsample of the training dataset. The output probabilities, or some intermediate feature representations, of each CNN, are then extracted from the original data and pooled together to form new features ready for the second level of classification. To make best use of the trained CNN models, we manage to partially recover the lost information due to spatial subsampling in the pooling layers when forming feature vectors. Performance of the ensemble methods are evaluated on BOSSbase by detecting S-UNIWARD at 0.4 bpp embedding rate. Results have indicated that both the recovery of the lost information, and learning from intermediate representation in CNNs instead of output probabilities, have led to performance improvement.

Keywords

Steganalysis; Forensics; Convolutional Neural Networks; Deep Learning.

1. INTRODUCTION

When performance is highly concerned, ensemble learning has been arguably one of the most widely adopted techniques to solve machine learning problems since the invention of boosting [1][2], bootstrap aggregation [3] and random forest [4]. It is well-known that the neural networks, including the convolutional neural networks (CNNs), which have achieved great success in the fields of computer vision [5-9], are suitable to serve as base learners and form ensembles. In computer vision, the most prominent research studies focus on designing efficient CNN architectures, and seeking ways to improve the optimization efficiency of deep neural networks [8][9]. Nevertheless, ultimate performance is always brought by ensembles of multiple CNNs, e.g., all the winning solutions in the ImageNet Large Scale Visual Recognition Challenge [6][7][9] from year 2012 to 2015, are ensembles of multiple CNNs.

IH&MMSec 2016, June 20-23, 2016, Vigo, Spain
© 2016 ACM. ISBN 978-1-4503-4290-2/16/06..$15.00
DOI: http://dx.doi.org/10.1145/2909827.2930798

Inspired and encouraged by the success of CNNs in computer vision, the forensics society have started devoting research efforts on migrating the CNNs to solve forensics and steganalysis problems [10-13]. In [10], the proposed CNN boosted accuracy on detecting median filtering processing in images by 1% ~ 8% compared with previous works. In [11] and [12], attempts were made in applying CNNs to image steganalysis, although the reported performance are still worse than the conventional state-of-the-art works [14][18-20]. All of those works reported results using only a single CNN for each individual experiment. In [15], the structural design of CNNs for steganalysis was discussed, and the ensemble (five CNNs) performance of the proposed CNN is competitive compared with that achieved by the SRM [14]. The ensemble method used is model averaging (averaging the output probabilities of each CNN). In this paper, we go beyond model averaging, and test the performance of second-level classifiers trained on the feature vectors generated from base learners (CNNs). The feature vectors come from pooling (1) the direct output probabilities generated from the trained CNNs; (2) the output probabilities generated from the CNNs with offsets in the spatial subsampling steps of pooling layers; (3) the output vectors of the convolutional modules in CNNs. The second one aims at recovering the information loss caused by spatial subsampling. The performance of all the proposed ensemble methods are evaluated on BOSSbase [16] by detecting S-UNIWARD [22] at 0.4 bpp embedding rate. Results have indicated that both the recovery of the lost information caused by subsampling, and learning from features representations within CNNs instead of output probabilities, have led to performance improvement. While only tested on one dataset with a special steganalysis problem, the proposed ensemble methods should be generically applicable to most of the image forensic tasks using CNNs.

The rest of this paper is organized as follows. In Section 2, we briefly review the CNN architecture used to build base learners. The ensemble methods we have studied are listed in Section 3. Experimental results and discussions are presented in Section 4. Conclusion is drawn in Section 5.

2. CNN: THE BASE LEARNER

2.1 The CNN Architecture

The CNN architecture used in this paper is almost same as that proposed in [15] except that we append one more group of layers (Group 6 in Figure 1) to the end of the convolutional module, and increase the pooling sizes of the last two pooling layers from 5×5 to 7×7. This paper aims at studying strategies for ensemble learning instead of designing CNNs. To make this paper self-

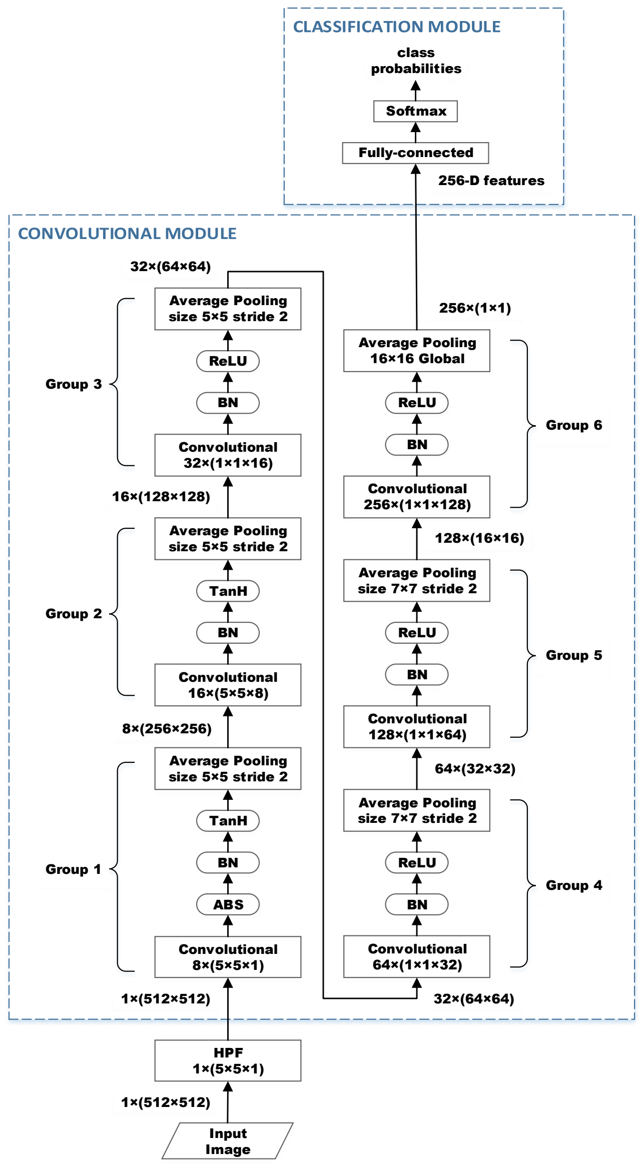

Figure 1. The CNN architecture. Inside boxes are the layer functions. Data sizes are displayed on the two sides. Sizes of convolution kernels in the boxes follow (number of kernels) × (height × width × number of input feature maps). Sizes of data follow (number of feature maps) × (height × width). Padding is applied in convolutional and pooling layers as well as in HPF layer.

contained, we briefly review the CNN architecture used for generating base learners. More details have been documented in [15].

The overall architecture is illustrated in Figure 1. A high-pass filtering (HPF) layer using the previously developed kernel [14] is placed at the very beginning to transform original images to noise residuals. The parameters in the HPF layer are not optimized during training, this CNN actually learns from the generated noise residuals instead of from the original images. Hence, in the rest of the paper, the training data refers to the obtained residuals from the original images. The whole CNN contains a convolutional module responsible to transform the images/residuals to feature vectors (256-D), which serves as input to the linear classification module that generates probability output for each of the two classes. Note that for binary classification problem, only one of the probability values is needed. The convolutional module comprises six groups of layers ("Group 1 ~ Group 6" in Figure 1), each of them starts with a convolutional layer, which doubles the number of spatial maps (often be referred to as the width of a layer in deep neural networks), and ends with an average pooling layer which performs local averaging as well as subsampling on the spatial maps (except Group 6). The CNN is equipped with the hyperbolic tangent (TanH) as non-linear activations for Group 1 and Group 2, and the rectified linear unit (ReLU) activations for Group 3 ~ Group 6. An absolute activation (ABS) layer is inserted in Group 1 to force the CNN to take into account the (sign) symmetry existed in noise residuals. Immediately before each non-linear activation layer, the feature maps are normalized with batch-normalization (BN) [8].

Through global averaging, the pooling layer in Group 6 merges each spatial map to a single element: 256 maps of size 16×16 to 256-D features. In this paper, we represent the size of the CNN by the output size of the last pooling layer (in Group 6), hence, we call the CNN in Figure 1 as 'SIZE 256', and 'SIZE 128' refers to a CNN with only half the width for each layer and roughly one quarter of the total number of parameters existed in 'SIZE 256'.

2.2 Ensemble Methods

In this section, we discuss in detail all the ensemble methods we have studied in this paper.

Let $\{X_i, y_i\}_{i=1}^{N}$ denote the training dataset, where N is the total number of training data – the residuals generated by the HPF layer, X_i represents the i-th residual of the training set, and y_i is the corresponding binary label. Note that, for $1 \leq i \leq N$, we have $X_i \epsilon \mathbf{R}^{h \times w}$ and $y_i \epsilon \{0,1\}$. A total of T CNNs are trained and used as base learners. In this work, we choose T = 16. The k-th CNN h_k maps each residual image to a probability value, and is represented by $h_k = h(X; \boldsymbol{w}_k) \colon \mathbf{R}^{h \times w} \to p\epsilon[0,1]$, in which the parameters $\boldsymbol{w}_k$ is optimized by minimizing the log loss denoted generally by $L_H = L(h, y)$. The procedures to generate base learners (CNNs) are summarized below:

```
1.  for $k = 1$ to $T$ do
2.      generate a random permutation $\{\pi(i)\}_{i=1}^{N}$ of $\{1,2,\dots,N\}$
3.      train $h_k$ specified by its parameters $\boldsymbol{w}_k$ :
        $\boldsymbol{w}_k = \underset{\boldsymbol{w}}{argmin}\ L_H\big(h(X_{\pi(i)}; \boldsymbol{w}), y_{\pi(i)}\big),\ i = 1, \dots, N^{tr}$
        $(N^{tr} < N)$
4.  end for
5.  collect all trained CNNs $\{h_k\}_{k=1}^{T}$
```

Note that this process is almost same as bootstrap aggregation [3], in which each base learner is trained on a subset randomly drawn by sampling with replacement from the original training set. In this work, sampling without replacement is used instead.

Once the training of CNNs is completed, the most straightforward and commonly used ensemble strategy is to average the output from each CNN and compare the result with th = 0.5 to determine the corresponding class label, i.e., for each test data X_t, the label $\hat{y}_t$ can be estimated by

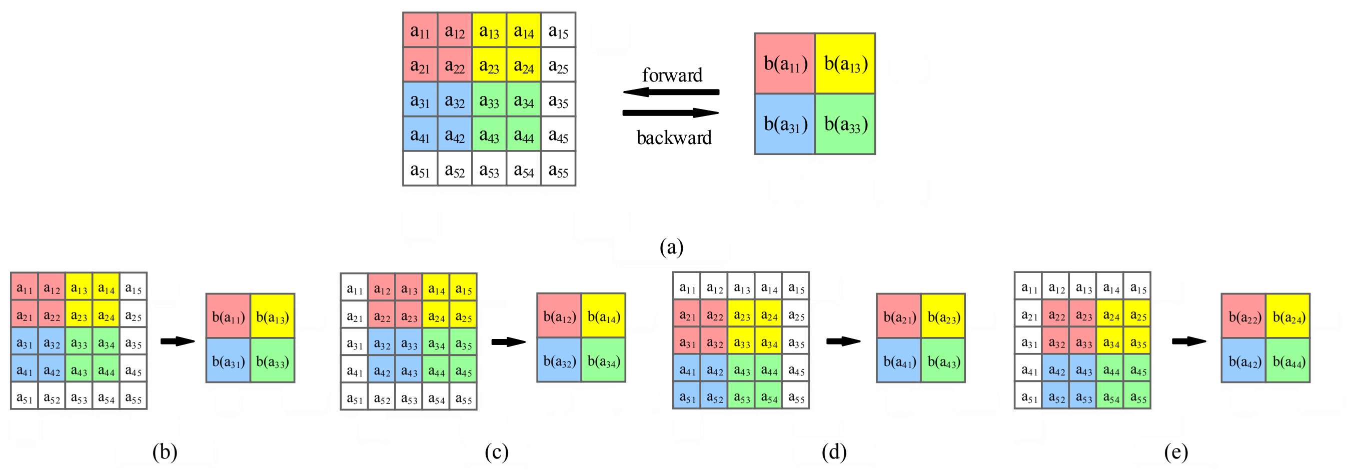

Figure 2. Pooling with local size 2×2 and stride 2 in horizontal and vertical direction. (a) Forward and backward passes of a pooling layer with fixed sampling locations in the training stage of the CNNs. (b) – (e) The four possible sampling when transforming image data with CNNs into feature vectors for ensemble learning.

$$\hat{y}_t = \begin{cases} 0, & \frac{1}{T}\sum_{1\le k\le T} h_k(X_t) < th \\ 1, & \frac{1}{T}\sum_{1\le k\le T} h_k(X_t) \ge th \end{cases}$$

This is the basic ensemble method performed in our experiments. Note that this basic model averaging strategy does not require further ensemble learning. Next, we will show that besides simple model averaging, more can be dug out from the CNNs for steganalysis tasks.

CNNs usually adopt several subsampling steps to reduce the spatial dimensions and facilitate classification. These subsampling steps are fulfilled in pooling layers or convolutional layers with strides set larger than 1. In computer vision and other related research areas, the subsampling steps may not have negative effect, because they discard irrelevant information and help the optimization in deeper layers focused. However, as steganalysis relies on statistics, spatial subsampling could cause information loss, even after the information of skipped pixels have been encoded into neighboring pixels through, e.g., averaging. The dilemma is, it seems that the spatial subsampling is unavoidable, because without it, the statistical modeling in CNN would grasp the location information of embedded pixels from the training data. To help alleviate this issue, one possible solution is, given a trained CNN, we generate probability output of every possible subsampling so that every skipped pixel location could be covered once, as illustrated in Figure 2 (b–e). To facilitate the explanation, we assume the pooling regions to be 2×2 and stride equals 2 for both horizontal and vertical direction. During training (Figure 2 (a)), the pooling layer sticks to only one set of spatial subsamples, i.e., a_{11}, a_{13}, a_{31}, a_{33}… The output of this pooling layer is calculated as, e.g., for average pooling, $b_{11} = (a_{11} + a_{12} + a_{21} + a_{22}) / 4$. But the skipped locations, e.g., $b_{12} = (a_{12} + a_{13} + a_{22} + a_{23}) / 4$ is never considered in training. The solution is to output probability values of all the four constellations of subsampling for ensembles, as demonstrated in Figure 2 (b–e). Following this, given P pooling layers in a CNN, and assume stride equals 2, the total number of output (probabilities) M generated from each trained CNN equals 4^P, for the CNN illustrated in Figure 1, we have $P = 5$ (in Group1 ~ Group5), and therefore $M = 1024$. In this scenario, using the averaging strategy, the class label $\hat{y}_t$ is estimated as:

$$\hat{y}_i = \begin{cases} 0, & \frac{1}{TM}\sum_{\substack{1\le k\le T\\ 1\le d\le M}} h_k^d(X_i) < th \\ 1, & \frac{1}{TM}\sum_{\substack{1\le k\le T\\ 1\le d\le M}} h_k^d(X_i) \ge th \end{cases}$$

One might realize that only 1 out of M probabilities is optimized during training for each CNN, the others are close to the optimized because of spatial correlations but are still suboptimal. In this case, it would be beneficial to map the original training data by the CNNs into a new feature representation and train second-level classifiers for optimal performance, as summarized:

1. map the original training data $\{X_i\}_{i=1}^N$ with base learners into $\{Z_i\}_{i=1}^N$, where $Z_i = \{h_k^1(X_i), h_k^2(X_i), \dots, h_k^M(X_i)\}_{k=1}^T$, for $1 \le i \le N$
2. build a classifier using $\{Z_i, y_i\}_{i=1}^N$, $i = 1, \dots, N$

In this paper, the ensemble classifiers [17] developed specifically for steganalysis is used as the second-level classifier because of its good performance and efficiency.

The last ensemble strategy we have tested is to gather from each CNN the output of the last pooling layer, which is also the output of the convolutional module and input of the classification module as displayed in Figure 1. The intuition is that the ensemble classifiers proposed in [17] are stronger compared with the logistic regression adopted in the classification modules of CNNs, and concatenating intermediate representations from every CNNs before performing classification potentially increases the chance of mining more discriminative patterns. Let f_k denote the function of the convolutional module in the k-th CNN. This ensemble method can be summarized as:

1. map the original training data $\{X_i\}_{i=1}^N$ with base learners into $\{Z_i\}_{i=1}^N$, where $Z_i = \{f_k(X_i)\}_{k=1}^T$, for $1 \le i \le N$
2. build a classifier using $\{Z_i, y_i\}_{i=1}^N$, $i = 1, \dots, N$

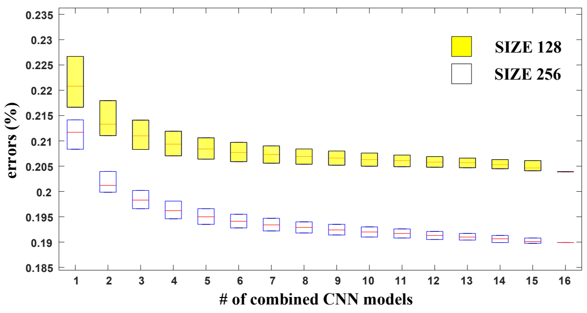

Figure 3. Box plots reflecting overall performance with different number of combined CNN models, for both 'SIZE 128' and 'SIZE 256'. Red lines are the median values, and the upper and lower bounds correspond to the 25 and 75 percentiles.

Table 1. Feature Dimensions of Different Ensemble Scenarios.

	SIZE 128		SIZE 256	
	Ensemble Methods		Ensemble Methods	
	AVE	**ENS**	**AVE**	**ENS**
PROB	16	16	16	16
PROB_POOL	16384	16384	16384	16384
FEA	N/A	2048	N/A	4096

Table 2. Detection Errors (%) of Different Ensemble Scenarios.

	SIZE 128		SIZE 256	
	Ensemble Methods		Ensemble Methods	
	AVE	**ENS**	**AVE**	**ENS**
PROB	0.2039	0.1973	0.1899	0.1897
PROB_POOL	0.2018	0.1954	0.1918	0.1871
FEA	N/A	0.1906	N/A	0.1844

3. EXPERIMENTS

3.1 Dataset and Settings

Training of the CNNs is performed on a modified version of Caffe toolbox [21]. Performance of the ensemble methods is evaluated by detecting S-UNIWARD [22] at 0.4 bpp embedding rate only, due to the long training time of CNNs and the long feature mapping time. It takes about three weeks to run all the experiments using two NVIDIA Geforce GTX 980Ti graphics cards. The dataset used is BOSSbase v1.01 [16] containing 10,000 cover images of size 512×512. Image data of the other class (stego) were generated through data embedding into the cover images. Hence, the dataset contains 10,000 pairs of images. Out of the 10,000 pairs of images, 5,000 pairs were set aside for testing to verify the performance; the rest 5,000 pairs were used as the training set. To train each CNN as base learner, 4,000 out of the 5,000 training data were randomly drawn, the rest 1,000 data were used as validation set to prevent the neural networks from overtraining. Two groups of CNNs with different network sizes are obtained: 'SIZE 256' and 'SIZE 128', the numbers refer to the output size of the convolutional module as explained in Section 2.1. A total of 16 CNNs are trained and used as base learners for both the two network sizes.

For reproducibility, information of the hyperparameters and settings used during training is summarized here. The learning rate was initialized to 0.001, and scheduled to decrease 10% for every 5,000 iterations. The momentum was set to 0.9. A mini-batch of 64 images (32 cover/stego pairs) was input for each iteration. All of the CNNs were trained for 120,000 iterations (960 epochs). Weight decay was not enabled except for the FC layers.

3.2 Results and Discussion

In the first experiment, we study how the number of CNNs used for ensemble affect the performance. For simplicity, the basic model averaging strategy is adopted. For every fixed number of CNNs used for ensemble, we tested all the combinations (out of 16), and record the box plot for both of the two networks sizes. From Figure 3, we can conclude that increasing the number of CNNs for ensemble reduces variance, and consistently reduces detection errors. Comparing the performance of the two networks with different sizes, we observe that 'SIZE 256' has both lower bias and lower variance, which indicates that the width of a CNN is very important for steganalysis.

Table 2 records all the results using the ensemble strategies proposed in Section 2.2. In Table 1, the number of total features for each ensemble scenario is presented. In Table 1 and Table 2, ***PROB*** refers to the direct CNN probability output. ***PROB_POOL*** refers to the generation of probabilities subsampling method and for each CNN. ***FEA*** corresponds to the output of the convolutional modules in CNNs. ***AVE*** means simple model averaging, and ***ENS*** is the ensemble classifiers [17]. From Table 2, we can summarize that the second-level learning consistently yields better performance compared with model averaging. When the ensemble learning method is fixed to ***AVE***, ***PROB_POOL*** does not always have better performance over ***PROB***, probably due to the suboptimal probabilities output discussed in Section 2.2. The best performance is always achieved by learning from the pooled output of the convolutional modules, which indicates that for performance, it might be preferred to abandon the classification modules in CNNs. To have some idea of where the presented ensemble performance are, the 34671-D SRM model [14] with ensemble classifiers [17] on the same train/test split, had an error rate of 0.2047.

4. CONCLUSION

In this paper, we study different ensemble strategies using CNNs as base learners for steganalysis. To make best use of the trained CNN models, we manage to partially recover the lost information due to spatial subsampling in the pooling layers when forming feature vectors for ensemble learning. Results suggest that both the recovery of the lost information caused by spatial subsampling, and learning from intermediate representation in CNNs instead of output probabilities, improve the performance. While only tested on one dataset with a special steganalysis, the proposed ensemble methods should be generic and could be performed in most of the image forensic tasks using CNNs.

5. REFERENCES

[1] Y. Freund, and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comp. Syst. Sciences*, 55(1): 119-139, Aug. 1997.

[2] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals Statistics*, 29(5): 1189-1232, Oct. 2001.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24(2): 123-140, Aug. 1996.

[4] L. Breiman. Random forests. *Machine learning*, 45(1): 5-32, Oct. 2001.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In: *Proc. Adv. Neural Inf. Process. Syst.*, pages 1097-1105, 2012.

[6] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-fei. Imagenet large scale visual recognition challenge. *Int. J. Comp. Vision*, 115(3): 211-252, Apr. 2015.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In: *Proc. IEEE Conf. Comp. Vision Pattern Recognition*, pages 1-9, 2015.

[8] S. Ioffe, and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. *arXiv*:1502.03167, Feb. 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv*:1512.03385, Dec. 2015.

[10] J. Chen, X. Kang, Y. Liu, and Z. J. Wang. Median filtering forensics based on convolutional neural networks. *IEEE Signal Process. Lett.,* 22(11): 1849-1853, Jun. 2015.

[11] S. Tan, and B. Li. Stacked convolutional auto-encoders for steganalysis of digital images. In: *Proc. APSIPA*, pages 1-4, Dec. 2014.

[12] Y. Qian, J. Dong, W. Wang, and T. Tan. Deep learning for steganalysis via convolutional neural networks. In: *Proc. SPIE Electronic Imaging*, pages 94090J, Mar. 2015.

[13] L. Pibre, P. Jérôme, 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 source-mismatch. In: *Proc. SPIE Electronic Imaging*, Feb. 2016.

[14] J. Fridrich, and J. Kodovský. Rich models for steganalysis of digital images. *IEEE Inf. Forensics Security*, 7(3): 868-882, Jun. 2012.

[15] G. Xu, H. Wu, and Y. Shi. Structural design of convolutional neural networks for steganalysis. *IEEE Signal Process. Lett.,* 23(5): 708-712, Mar. 2016.

[16] P. Bas, T. Filler, and T. Pevný. "Break our steganographic system": the ins and outs of organizing BOSS. In: *Proc. Inf. Hiding*, vol. 6958, pages 59-70, Springer, May 2011.

[17] J. Kodovsky, J. Fridrich, and V. Holub. Ensemble classifiers for steganalysis of digital media. *IEEE Trans. Inf. Forensics Security,* 7(2): 432-444, Apr. 2012.

[18] V. Holub, and J. Fridrich. Random projections of residuals for digital image steganalysis. *IEEE Trans. Inf. Forensics Security*, 8(12): 1996-2006, Dec. 2013.

[19] T. Denemark, V. Sedighi, V. Holub, R. Cogranne, and J. Fridrich. Selection-channel-aware rich model for steganalysis of digital images. In: *Proc. Inf. Forensics Security (WIFS)*, pages 48-53, Dec. 2014.

[20] W. Tang, H. Li, W. Luo, and J. Huang. Adaptive steganalysis against WOW embedding algorithm. In: *Proc. ACM IH&MMSec*, pages 91-96, 2014.

[21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: convolutional architecture for fast feature embedding. In: *Proc. ACM Int. Conf. Multimed.*, pages 675-678, 2014.

[22] V. Holub, J. Fridrich, and T. Denemark. Universal distortion function for steganography in an arbitrary domain. *EURASIP J. Inf. Security*, 2014(1): 1-13, Dec. 2014.