https://doi.org/10.2352/ISSN.2470-1173.2017.7.MWSF-325

Histogram Layer, Moving Convolutional Neural Networks Towards Feature-Based Steganalysis

Vahid Sedighi and Jessica Fridrich, Department of ECE, SUNY Binghamton, NY, USA, {vsedigh1,fridrich}@binghamton.edu

Abstract

Feature-based steganalysis has been an integral tool for detecting the presence of steganography in communication channels for a long time. In this paper, we explore the possibility to utilize powerful optimization algorithms available in convolutional neural network packages to optimize the design of rich features. To this end, we implemented a new layer that simulates the formation of histograms from truncated and quantized noise residuals computed by convolution. Our goal is to show the potential to compactify and further optimize existing features, such as the projection spatial rich model (PSRM).

Motivation

In steganography, the main goal is to communicate secretly through an overt channel. Steganalysis, on the other hand tries to detect the presence of steganography. For a long time, detection relied on classifiers trained in a supervised fashion on examples of cover and stego images [3]. To detect content-adaptive schemes, researchers hand crafted various high-dimensional feature representations (rich models). Such features are typically formed from noise residuals extracted from the input image by convolutions with high-pass linear filters (kernels). The array of residuals would then be represented either with co-occurrence matrices (empirical joint densities) [19, 12, 2] or via histograms of projections on random directions [6]. Due to the inherent complexity of digital images, the design of suitable kernels has been based entirely on heuristics. While there were attempts to optimize the kernels by parametrizing them and determining the parameters by minimizing the classifier detection error using, e.g., the Nelder–Mead algorithm [5], the complexity of evaluating the objective function (training a classifier on thousands of images and evaluating its performance using, e.g., the minimal total error probability P_{E}) makes such approaches non-scalable and unable to optimize a sufficient number of kernels to build a rich model. The main flaw is the need for a constant feedback from the classifier, which is very time consuming due to the complexity of training with high-dimensional features on thousands of images.

The main advantage of Convolutional Neural Networks (CNNs) is their ability to optimize the feature extraction and classification steps simultaneously and thus close the loop between feature extraction and classification. Such networks are not only capable of learning the best decision boundary between different classes but also the best representation of each data class that would improve their separability. Originally developed for numerous computer vision problems, these networks were recently adapted for steganalysis [16, 14, 13, 17, 18]. Since in a typical computer vision problem, CNNs are used to learn patterns or objects, in steganalysis the signal of interest is hidden within the noise component of the image. In order to deal with this issue, in one of the early works in this direction Qian et al. [14] proposed to adjust the network design by using one of the successfully hand-designed high-pass kernels in the Spatial Rich Model (SRM) [3] as the first (fixed) convolutional layer in the network. The main reason behind this step is to suppress most of each image content and thus force the network to "pay attention" to high-frequency details.

A closer look at the CNN structure reveals a similarity with feature-based steganalysis. The convolutional layers play the role of residual extraction while activation and pooling layers mimick truncation and quantization. Various researches tried to improve the performance of CNNs for steganalysis using insights acquired from feature-based steganalysis. In one of the most recent works in this direction, Xu et al. [17] proposed a novel CNN architecture capable of approaching the performance of feature-based steganalysis. In their five-layer CNN, after initial high-pass filtering and the first convolutional layer, they proposed to take the absolute value (ABS) of the feature maps (residuals) as the activation function followed by a hyperbolic tangent (TanH) in order to preserve the sign symmetry of the residuals in a process similar to computing SRM residuals. Additionally, by using 1×1 convolutional kernels in the last three layers, the authors forced the network to collect the local statistics from the feature maps in a pixel by pixel fashion.

Inspired by similarities between conventional feature-based and CNN detectors, in this paper we investigate the possibility to use existing infrastructure behind CNNs to optimize the design of kernels (linear pixel predictors) in feature-based steganalysis. The replacement of the objective function in the form of some scalar classifier performance criterion by the loss function in a CNN, powerful gradient descend algorithms could be used for the optimization task provided there was a way to form histograms within a CNN. Thus, as our first step, we implemented a histogram layer for the Caffe CNN package. We use mean-shifted Gaussian functions as the building blocks of this layer to obtain a proper back-flow of gradients through the layer and to facilitate learning of the parameters behind this layer. As a proof of concept, in this paper we use this layer to model individual submodels of the PSRM [6]. Instead of forming co-occurrences of neighboring quantized

residual samples, PSRM projects unquantized residual values on random directions, which are subsequently quantized and represented using histograms as steganalysis features. In PSRM, each SRM kernel is projected on 55 random two-dimensional kernels and their rotated and mirrored versions. The higher detection rate of this feature set is the result of a large number of projections, which comes at a high computational cost. This problem can render this powerful feature set unusable for applications with limited time and computational power [10]. Modeling these submodels within the CNN framework with the histogram layer enables us to reduce the high dimensionality of this feature set by replacing random kernels with fewer optimized kernels. Our study also hints at the possibility to extract more information in the final layers of CNNs to pave the way for better network designs.

In the next section, we introduce the histogram layer and discuss its design and internal building blocks. In Section "Experimental setup," we simulate PSRM models within the CNN framework using the histogram layer and outline the training procedure. The proposed histogram layer is tested in Section "Analyzing results," where we compare the detection results of our CNN model with PSRM submodels and discuss the results. The paper is summarized in the last section.

Histogram layer

To capture the local statistics of feature maps using histograms without any information loss we need to use step functions centered on each histogram bin. While shifted step functions are the best choice to compute independent histogram bins, they are unusable within a CNN framework because their derivative is zero everywhere except for the edges. This prevents the back flow of gradients through the layer and the back-propagation algorithm stops working. In contrast, a Gaussian kernel seems to be a good candidate to form histogram bins. Unlike the sharp edges of a step function, its smooth slopes will create a fine path for gradients to flow backwards through each histogram bin towards previous layers. A Gaussian activation function has the form

$$g(x) = e^{-\frac{(x-\mu)^2}{\sigma^2}} \tag{1}$$

where μ is the center of the histogram bin and σ controls the tradeoff between the accuracy of each binning operation (and the overlap between adjacent bins) and the flow of gradients through the layer. In our experiments, we fixed $\sigma = 0.6$ to obtain a close match between the exact histogram and a histogram computed using a Gaussian activation function. Fig. 1 shows the structure of an 8-bin histogram function using Gaussian activations. The value of μ for each kernel is chosen from the set $\mu \in \{-3.5, -2.5, \ldots, 2.5, 3.5\}$. The tails of the Gaussian kernels on the sides are replaced with a constant value of 1 in order to simulate the residual truncation done in feature-based steganalysis. To compute the value of the histogram bin $B(k)$ centered at $\mu = \mu_k$ for an $M \times N$ activation map

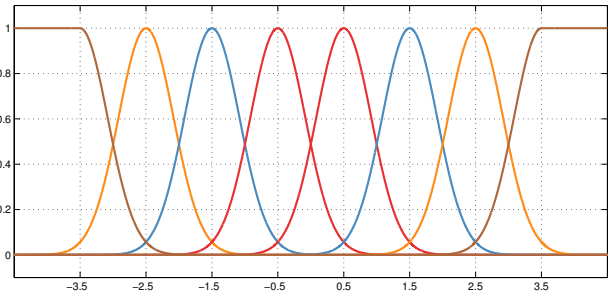

Figure 1. *Structure of an 8-bin histogram using mean-shifted Gaussian kernels.*

X, all of its elements x_{ij} must pass through the corresponding Gaussian activation and get normalized:

$$B(k) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} e^{-\frac{(x_{ij}-\mu_k)^2}{\sigma^2}}. \tag{2}$$

Using 2, each feature map X will be represented by K histogram bins $B(k)$, $k = 1, 2, \ldots, K$, where K is the number of histogram bins. We set the value of $K = 8$ in our experiments to match PSRM histograms. All computed histograms from each feature map will be concatenated as a vector and passed to the last fully connected layers of the CNN for classification.

During back propagation, the gradient of the loss function L with respect to each element of the feature maps x_{ij} will be computed using the following equation,

$$\frac{\partial L}{\partial x_{ij}} = \frac{1}{MN} \sum_{k=1}^{K} \frac{-2(x_{ij}-\mu_k)}{\sigma^2} e^{-\frac{(x_{ij}-\mu_k)^2}{\sigma^2}} \frac{\partial L}{\partial B(k)} \tag{3}$$

where $\frac{\partial L}{\partial B(k)}$ is the gradient of the loss function with respect to each histogram bin $B(k)$. Clearly, the gradient will be zero when $x_{ij} > \mu_k$ in the rightmost kernels and when $x_{ij} < \mu_k$ in the leftmost kernels depicted in Fig. 1 with brown color on the sides.

Our histogram layer is using C++ and CUDA in both CPU and GPU versions within Caffe [9] framework and optimized for memory usage and speed.

Experimental setup

To evaluate the performance of our proposed histogram layer, we model one of the most effective submodels of PSRM using the histogram layer within a CNN framework. Fig. 2 depicts the general block diagram of this powerful feature set.

In the first stage, the PSRM computes the residuals with two-dimensional convolutions between the input image and all SRM kernels. For simplicity, in this study we will only focus on one of the most effective SRM kernels shown in Fig. 3 called 'SQUARE5x5' also known as the KV kernel. The goal of this residual extraction step is to suppress the content while preserving the stego noise. In

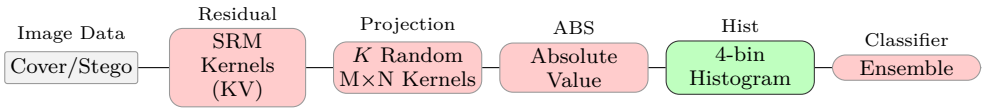

Figure 2. *Block diagram of PSRM sub-models. The size of the random projection kernels is randomly chosen from the set* $M, N \in \{1, 2, 3, ..., L = 8\}$.

the second step, the extracted residuals are projected on K random $M \times N$ kernels. Each random kernel will be generated from a zero mean unit variance Gaussian distribution, $\mathcal{N}(0, 1)$, and the projection kernel's dimensionality is randomly chosen from the set $M, N \in \{1, 2, 3, ..., L = 8\}$. Next, each residual goes through the absolute value function and, after quantization, 4-bin histograms will collect local statistics of each projected residual. These histograms will form feature vectors that will determine whether the input image is cover or stego. During the projection step, if the random size of the projection kernels allow, each residual will also get projected onto mirrored and rotated versions of the kernels and all of the resulting projected values will contribute to the corresponding histogram bins of each particular kernel. This can potentially increase the number of projections of each SRM residual up to $4K$. Finally, due to high dimensionality of the computed feature sets, the FLD-ensemble classifier [11] will be used to form the decision boundary between cover and stego classes to assess the performance of the computed features.

$$\mathrm{KV} = \begin{pmatrix} -1 & 2 & -2 & 2 & 1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & 1 \end{pmatrix}$$

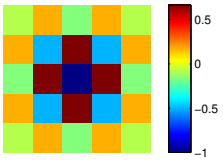

Figure 3. *Matrix form of the KV kernel from the SRM (left) and its normalized plot (right).*

Having the convolutions as standard modules in a CNN framework, we can easily model the first two blocks of the PSRM diagram, computing residual and projections, within this framework. Fig. 4 illustrates different sections of our modeled version of PSRM submodels.

Since the focus of this study is only the KV kernel, in the first layer we use a single 5×5 convolution kernel and initialize the weights with the KV kernel and a zero bias. To prevent the learning process to change the filter parameters, the learning rate multiplicative coefficient of this layer for both kernel weights and bias is set to zero.

In the next layer, we learn K different convolution kernels of size $M \times M$ randomly initialized using a Gaussian distribution $\mathcal{N}(0, 0.01)$. In order to be consistent with the PSRM setup, we initialize the bias term of the kernels to zero and also set their learning rate multiplicative coefficient to zero. The original version of the PSRM uses up to $K = 55$ projection kernels per each SRM residual and allows the size of each kernel to be random from $\{1, \ldots, L\}$, $L = 8$. For a fair comparison between both approaches, we thus set the total number of PSRM kernels to $K = 32$, the maximum number of kernels trained in our CNN model. Also, due to limitation of having to have odd numbers for kernel sizes in convolution modules of the CNN, we set $L = 7$ and fixed the CNN kernel's dimension at $M = 7$. It is worth noting that while PSRM enjoys having more diverse kernel size dimension from the set $\{1, \ldots, 7\}$, the CNN model is limited to training kernels of the same dimension set at 7×7. We study the effect of this limitation on the kernel size in more detail in the "analyzing results" section.

To increase the diversity of the trained kernels, we allow the network to learn asymmetric kernels by not passing the projected residuals through the absolute value function and directly collecting statistics using our proposed histogram layer with 8 bins. This can be interpreted as an unfolded version of the absolute value and a four-bin histogram in the PSRM. Finally, to close the training loop we use a two-layer neural network as our classifier on top of the collected histograms. The first fully connected layer has 10 and the last layer has 2 neurons representing the cover and stego classes. All neurons are initialized randomly using $\mathcal{N}(0, 0.01)$ and the biases are set to a constant value of 0.2. We use the ReLU non-linearity in between the two classifier layers to allow for a decision boundary with more degrees of freedom. All neurons' weights and biases are getting updates during training with biases using twice the network's learning rate in general.

The Caffe framework is used to implement the CNN. We use Stochastic Gradient Descent (SGD) to minimize the loss function with mini batch size set to 64, momentum 0.98, and weight decay 0.0005. All trained weights in the network are regularized during training using the same rate. The learning rate is changed adaptively from 0.01 to 0.0001 during 2000 epochs of training. Using a step function, every 50 epochs we multiply the initial learning rate by 0.9. To find more general solutions during the learning process, we shuffle the training set after each epoch while preserving the pairs of cover and stego images inside each batch. All models are developed and trained using Tesla K80 and TITAN X GPUs from NVIDIA.

All experiments were carried out on BOSSbase 1.01 [1]. We used S-UNIWARD [8] as our embedding algorithm with relative payload fixed at $\alpha = 0.4$ bpp (bits per pixel). The FLD-ensemble classifier was used for classification of the extracted features. The ensemble by default minimizes the total classification error probability under equal priors $P_{\mathrm{E}} = \min_{P_{\mathrm{FA}}} \frac{1}{2}(P_{\mathrm{FA}} + P_{\mathrm{MD}})$, where P_{FA} and P_{MD} are the false-alarm and missed-detection probabilities. We evaluate the security by computing P_{E} measured on the testing set over a single random 5000/5000 database split. The same split is also used to train and test the CNN.

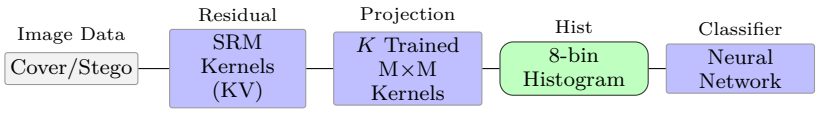

Figure 4. *Block diagram of Modeled PSRM sub-model within CNN framework. The dimensionality of the trained projection kernels is set to $M = 7$ in our experiments.*

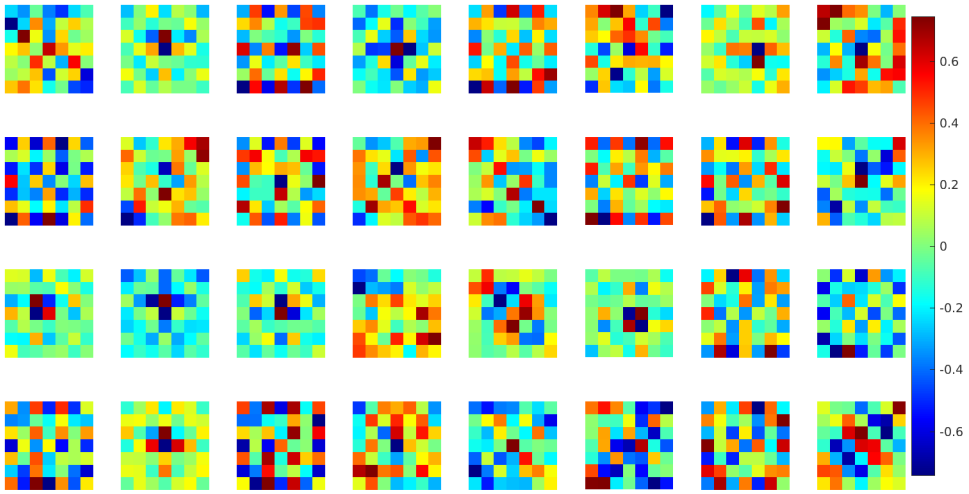

Figure 5. *The 32 trained PSRM kernels using a CNN with the histogram layer.*

Analyzing results

In our first experiment, we trained 32 projection kernels using the above version of the PSRM modeled within the CNN framework. Fig. 5 depicts various trained kernels using this model. The detection error of the trained model on the test set is $P_{\text{E}} = 27\%$.

In the next step, in order to asses the performance of the trained filters using our CNN model within the PSRM framework, we carve out the filters of the trained model, plug them into the PSRM, and use them instead of random kernels. In order to visualize the contribution of each kernel to total testing error, after extracting all features from the test set, we start steganalyzing only the features resulting from the first kernel (four histogram bins) in the beginning and then gradually add the features from the remaining kernels one by one inside our feature pool and repeat the steganalysis until we exhaust all available features. Since the dimensionality of the features from projections of only KV is small and since we want to study the effect of adding features from each kernel, we force the dimensionality of each subspace (d_{sub}) in the ensemble classifier to be equal the total size of the feature vector in each step.

Fig. 6 contrasts the performance of the trained 32 kernels with the random kernels using two different setups. In PSRM, random kernels are allowed to have a variable dimension $M \times N$ from the set $\{1, \ldots, 7\}$ whereas in the CNN all random kernels' dimensions are fixed to 7×7, which makes the test condition fair with respect to the trained kernels.

The first interesting result that we can see from this diagram is the fact that the test performance of the trained filters within the PSRM model using the ensemble classifier is almost the same as the test performance of the filters within the CNN framework using the neural network classifier. This verifies the correctness of our model. We also see the importance of diversity among kernels' dimensions. The difference between the final performance of variable-size and fixed-size random kernels is almost 1.5%. The trained kernels beat the performance of the random kernels within the PSRM setup up to the 17th kernel. Beyond that, however, random kernels become consistently $\approx 1.5\%$ better up to the last one. Equalizing the experimental setup, the trained kernels beat the random ones almost constantly up to the last kernel by a narrow margin, and in the end the random kernels perform slightly better.

Even though the performance of the trained kernels under equal testing conditions is slightly better than the random kernels, one would expect the difference between the performance of the trained and random kernels to be larger due to a more intelligent design. This made us revisit the structure of the PSRM and analyze it in more detail. During the process of projection and binning of the computed residuals, in order to have a better population of histogram bins, the PSRM not only uses each random kernel itself but also its rotated and mirrored versions for collecting local statistics. This can potentially increase the number of projection kernels by a factor of four using random kernels and collect more powerful statistics from the projections. However, for the trained kernels the situation is slightly different. Since during the training process all images have passed through a fixed KV filter in the first

layer, the network's solution space for the trained kernels is limited. In other words, the SGD is seeing the solution space "through the eyes" of the fixed KV filter in the beginning and that can possibly prevent it from finding a better solution, hindering thus to some degree the training process. Inspecting the trained kernels in Fig. 5, we can easily see traces of the symmetry and the general checkerboard structure of the KV filter in most of the trained filters. This will work against the trained filters during the performance comparison with random kernels as rotating and mirroring these kernels will not extract new information from the residuals than the original version of the kernel. We explored the correctness of this hypothesis in our second experiment in which we only used the main kernels for projection and binning without adding the information from mirrored and rotated versions inside the corresponding histogram bins. Fig. 7 summarizes the result of this experiment.

We can clearly see that while this change does not noticeably impact the performance of the 32 trained kernels, it increases the test error of the random kernels by a considerable margin of 1.5%. At this point, the effectiveness of the training process becomes more evident.

We also trained three more models with fewer training kernels of 16, 8, and 4. Both models trained with 16 and 8 kernles can still achieve better test accuracy than 32 random kernels. Being able to achieve a close performance with fewer filters hints at a possible redundancy in the trained filters and thus a space for improving the training strategy.

Conclusions and future work

Convolutional Neural Networks are slowly but surely finding their way into different areas of signal processing by solving difficult problems with brute-forcing the solution space in an intelligent way. Recently, steganalysts have shown interest in these networks and their structure and tried to take advantage of having a closed loop between feature extraction and classification modules of CNNs for steganalysis. Closing this loop enables steganalyst for the

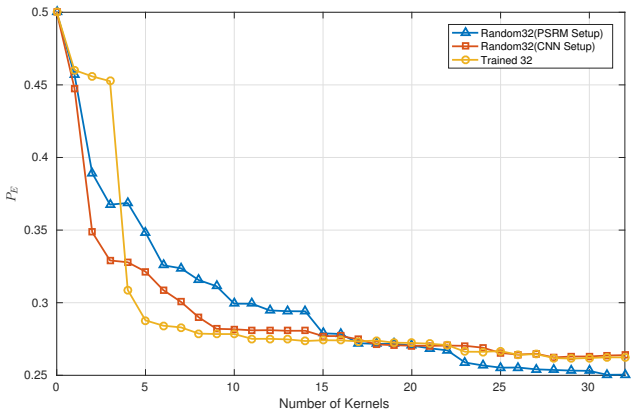

Figure 6. *Comparing the performance of the trained kernels versus random ones within the PSRM and the CNN experimental setup conditions.*

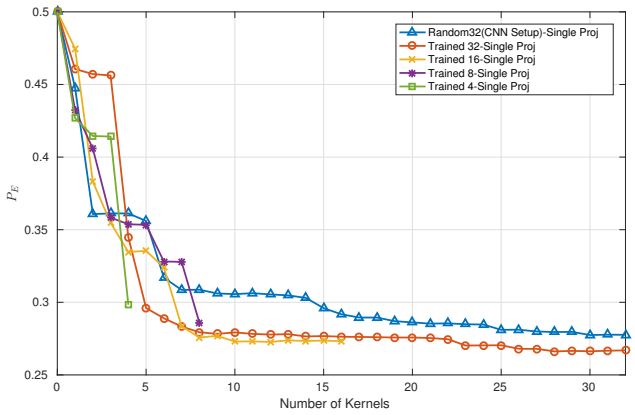

Figure 7. *Comparing the performance of the trained kernels from four different sizes* $\{4, 8, 16, 32\}$ *versus random ones within our CNN experimental setup conditions while using only the main kernels for projections and histogram binning.*

first time to use the training data for designing effective and robust feature extraction kernels. Due to the difference between the nature of the signal of interest in steganalysis and computer vision, researchers have tried to make the CNN network structure resemble conventional feature-based steganalysis in order to make them effective for this task. To the best knowledge of the authors, all CNN architectures proposed for steganalysis tasks so far have used elements of hand design, such as fixed convolution kernels in the first layer. We can also find traces of mimicking other modules in conventional steganalysis upon further inspection of these networks.

In this study, in order to bring these two designs even closer to each other we modeled one of the central concepts of feature-based steganalysis, the histograms, within a CNN framework in a way that allows proper training of the convolutional kernels in the first layers of the network. Using the proposed histogram layer, we modeled one of the most powerful feature sets for steganalysis, PSRM, within a CNN framework and showed that under an experimental setup similar to our model we are able to reduce the number of projection kernels by up to a factor of 4 by using optimized trained filters instead of random ones. This suggests that there it may be possible to significantly reduce the high dimensionality of the PSRM using a set of properly trained kernels while preserving its detection performance and simple design. Our work has not reached state-of-the-art performance due to some limitations in our modeling but it clearly is a proof of concept.

In our future work, we plan to enhance our model design and allow the histogram layer to learn the μ and σ parameters of the Gaussian functions during the training process from the data. This would enable us to optimize residual quantization and binning and also possibly bring useful insights for feature-based steganalysis. The histogram layer can also be used to model other feature sets formed as histograms of noise residuals obtained using convolutions, such as the DCTR [4], GFR [15], and PHARM [7]. Additionally, the istogram layer could be implanted in deeper

networks before the fully connected "classifier layers" to see if using this interesting element from conventional steganalysis could bring further improvements. Finally, the histogram layer could also be used to form co-occurrences from feature maps in order to collect alternative higher-order statistics.

The histogram layer and the config files for the CNNs used in this paper are available from `http://dde.binghamton.edu/download/`.

References

[1] 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.

[2] L. Chen, Y.Q. Shi, P. Sutthiwan, and X. Niu. A novel mapping scheme for steganalysis. In Y.Q. Shi, H.-J. Kim, and F. Perez-Gonzalez, editors, *International Workshop on Digital Forensics and Watermaking*, volume 7809 of *LNCS*, pages 19–33. Springer Berlin Heidelberg, 2013.

[3] 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.

[4] V. Holub and J. Fridrich. Low complexity features for JPEG steganalysis using undecimated DCT. *IEEE Transactions on Information Forensics and Security*. Under review.

[5] V. Holub and J. Fridrich. Optimizing pixel predictors for steganalysis. In A. Alattar, N. D. Memon, and E. J. Delp, editors, *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2012*, volume 8303, pages 09–1–09–13, San Francisco, CA, January 23–26, 2012.

[6] V. Holub and J. Fridrich. Random projections of residuals for digital image steganalysis. *IEEE Transactions on Information Forensics and Security*, 8(12):1996–2006, December 2013.

[7] V. Holub and J. Fridrich. Phase-aware projection model for steganalysis of JPEG images. In A. Alattar, N. D. Memon, and C. Heitzenrater, editors, *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2015*, volume 9409, San Francisco, CA, February 8–12, 2015.

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

[9] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *CoRR*, abs/1408.5093, 2014.

[10] A. D. Ker. Implementing the projected spatial rich features on a GPU. In A. Alattar, N. D. Memon, and C. Heitzenrater, editors, *Proceedings SPIE, Electronic Imaging, Media Watermarking, Security, and Forensics 2014*, volume 9028, pages 1801–1810, San Francisco, CA, February 3–5, 2014.

[11] 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.

[12] T. Pevný, P. Bas, and J. Fridrich. Steganalysis by subtractive pixel adjacency matrix. *IEEE Transactions on Information Forensics and Security*, 5(2):215–224, June 2010.

[13] Lionel Pibre, Pasquet Jérôme, Dino Ienco, and Marc 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 *Electronic Imaging*, Media Watermarking, Security, and Forensics, San Fransisco, United States, February 2016. SPIE.

[14] Yinlong Qian, Jing Dong, Wei Wang, and Tieniu Tan. Deep learning for steganalysis via convolutional neural networks, 2015.

[15] X. Song, F. Liu, C. Yang, X. Luo, and Y. Zhang. Steganalysis of adaptive JPEG steganography using 2D Gabor filters. In P. Comesana, J. Fridrich, and A. Alattar, editors, *3rd ACM IH&MMSec. Workshop*, Portland, Oregon, June 17–19, 2015.

[16] S. Tan and B. Li. Stacked convolutional auto-encoders for steganalysis of digital images. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, pages 1–4, Dec 2014.

[17] 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.

[18] Guanshuo Xu, Han-Zhou Wu, and Yun Q.. Shi. Ensemble of cnns for steganalysis: An empirical study. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '16, pages 103–107, New York, NY, USA, 2016. ACM.

[19] D. Zou, Y. Q. Shi, W. Su, and G. Xuan. Steganalysis based on Markov model of thresholded prediction-error image. In *Proceedings IEEE, International Conference on Multimedia and Expo*, pages 1365–1368, Toronto, Canada, July 9–12, 2006.

Author Biography

Vahid Sedighi received his B.S. degree in Electrical Engineering in 2005 from Shahed University, Tehran, Iran, and his M.S. degree in Electrical Engineering in 2010 from Yazd University, Yazd, Iran. He is currently pursuing the Ph.D degree in the Department of Electrical and Computer Engineering at Binghamton University, State University of New York. His research interests include statistical signal processing, steganography, steganalysis, and machine learning.

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 forensic. Since 1995, she has received 20 research grants totaling over $11 mil that lead to more than 180 papers and 7 US patents.