Structural Design of Convolutional Neural Networks for Steganalysis

Guanshuo Xu, Han-Zhou Wu, *Student Member, IEEE*, and Yun-Qing Shi, *Fellow, IEEE*

***Abstract*—Recent studies have indicated that the architectures of convolutional neural networks (CNNs) tailored for computer vision may not be best suited to image steganalysis. In this letter, we report a CNN architecture that takes into account knowledge of steganalysis. In the detailed architecture, we take absolute values of elements in the feature maps generated from the first convolutional layer to facilitate and improve statistical modeling in the subsequent layers; to prevent overfitting, we constrain the range of data values with the saturation regions of *hyperbolic tangent* (*TanH*) at early stages of the networks and reduce the strength of modeling using 1×1 convolutions in deeper layers. Although it learns from only one type of noise residual, the proposed CNN is competitive in terms of detection performance compared with the SRM with ensemble classifiers on the BOSSbase for detecting S-UNIWARD and HILL. The results have implied that well-designed CNNs have the potential to provide a better detection performance in the future.**

***Index Terms*—Convolutional neural networks (CNNs), deep learning, forensics, steganalysis.**

I. Introduction

THE FEATURE design in conventional machine learning-based steganalysis [1]–[6] could be suboptimal when the data embedding methods are effective and sophisticated [7]–[12]. Strong in feature learning, the convolutional neural networks (CNNs) [13], [14] have made tremendous achievements in computer vision since 2012 [14]–[18]. This has aroused the interest of researchers in the field of information forensics, in particular, on steganalysis, in seeking the way to use CNNs for steganalysis [19]–[21], [26].

In their pioneering work, Tan and Li [20] proposed a CNN which comprises three stages of alternating convolutional and max pooling layers, and sigmoid nonlinear activations. When detecting HUGO [7] at embedding rate of 0.4 bits per pixel (bpp) on the BOSSbase [22], the CNN achieved the error rate of 48% with random parameter initialization; after involving a high-pass kernel [3], [23] into parameter initialization of the first convolutional layer, and pretraining all of the parameters with unsupervised learning, they managed to reduce the error rate to 31%, still far away from the error rate of 14% achieved by the SRM [3] with ensemble classifiers [25].

A few months later, Qian *et al.* [21] presented a CNN equipped with a high-pass filtering (HPF) layer, Gaussian nonlinear activations, and average pooling for steganalysis. The reported detection error rates are 2%–5% higher than those achieved by the SRM on the BOSSbase with images downsized from 512×512 to 256×256, when detecting HUGO [7], WOW [8], and S-UNIWARD [9]. This is a significant boost in performance, but it is still inferior to the SRM.

Studies in these two pieces of works have indicated that taking into account the (domain) knowledge in steganography and steganalysis (e.g., using high-pass kernels to generate noise residuals) improves the detection performance of the CNNs. In conventional steganalysis, the domain knowledge is embedded in the feature extraction. Analogously, as the CNNs embrace the feature extraction step into the networks, the domain knowledge should be reflected in the network architectures.

Along this direction, in this letter, we propose a CNN that tries to incorporate the knowledge of steganalysis. In the detailed architecture, we take absolute values of elements in the feature maps generated from the first convolutional layer to facilitate and improve statistical modeling in the subsequent layers; to prevent overfitting, we constrain the range of element values at early stages of the networks and reduce the strength of modeling using 1×1 convolutions in deeper layers; besides, as have been proved effective in the previous works [20], [21], [26], the proposed CNN learns from noise residuals and uses average pooling. Although the proposed CNN is neither large nor deep and currently learns from only one type of noise residual, the experimental results have verified that its performance is comparable with that of the SRM with ensemble classifiers. This initial-stage work has confirmed that deep learning with CNNs is indeed a powerful machine learning tool for steganalysis. The results have also implied that a well-designed CNN would have the potential to provide a better detection performance when compared with the conventional machine learning-based steganalysis.

II. Proposed CNN

In this section, the overall architecture of the proposed CNN is introduced first. Then, the design in the CNN to improve statistical modeling is presented. Finally, we discuss about our design considerations on the nonlinear activations and spatial sizes of convolutional kernels.

Manuscript received January 25, 2016; revised March 16, 2016; accepted March 18, 2016. Date of publication March 29, 2016; date of current version April 14, 2016. This work was supported by NJIT Faculty Seed Grant Initiative. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jiwu Huang.

G. Xu and Y.-Q. Shi are with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: gx3@njit.edu; shi@njit.edu).

H.-Z. Wu is with the Southwest Jiaotong University, Chengdu 611756, China, and also with New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: h.wu.phd@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LSP.2016.2548421

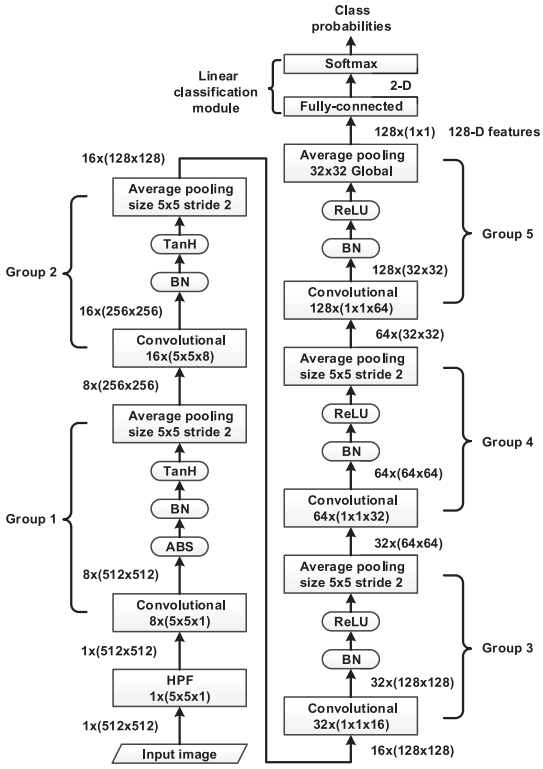

Fig. 1. Proposed CNN architecture. Layer types and parameter sizes (choices) are displayed inside boxes. Data sizes are displayed on the two sides. Sizes of convolution kernels in the boxes follow (number of kernels) × (height × width × number of input feature maps). Sizes of data follow (number of feature maps) × (height × width). Padding is applied in convolutional and pooling layers as well as the HPF layer.

A. Overall Architecture

Fig. 1 illustrates the overall architecture of our CNN. An HPF layer with the same kernel employed in [21] and [26], whose parameters are not optimized during training, is placed at the very beginning to transform original images to noise residuals to boost the signal–to-noise/interference ratio.

The whole CNN can be divided into a convolutional module which transforms the images to feature vectors (128-D), and a linear classification module (composed of a fully connected (FC) layer and a softmax layer) which transforms feature vectors to output probabilities for each class.

The convolutional module comprises five groups of layers (displayed as "Group 1" to "Group 5" in Fig. 1), each starts with a convolutional layer which generates feature maps, and ends with an average pooling layer which performs local averaging as well as subsampling on the feature maps (except Group 5).

To enhance the power of statistical modeling, our CNN is equipped with the hyperbolic tangent (*TanH*) [Fig. 3(a)] nonlinear activations for Group 1 and Group 2, and the rectified linear unit (ReLU) [24] activations [Fig. 3(b)] for Group 3, Group 4, and Group 5. Inside Group 1, an absolute activation (ABS) layer is inserted to force the statistical modeling to take into account the (sign) symmetry [3] existed in noise residuals. To prevent the CNN training from falling into poor local minima, and learn optimal scales and biases for feature maps, batch normalization (BN) [17] is performed immediately before each nonlinear activation layer.

Finally, through global averaging, the pooling layer in Group 5 merges each spatial map to a single element (128 maps of size 32×32 to 128-D features). In this way, the whole CNN is constraint to perform the same operations to every pixel in the original images (or the noise residuals), thereby preventing the statistical modeling from grasping the location information of embedded pixels from the training data. Note that the recent work [26] addresses the steganalysis scenario when the embedding key is reused for all the images, resulting in leakage of embedding locations. Hence, it differs in methodology with the work reported here.

B. Improving Statistical Modeling

The exact modeling procedure in the CNN is hard to interpret when the layerwise computation goes deeply. Therefore, we stand a better chance to improve the performance by focusing more the design of the first-layer group (Group 1).

Group 1 starts with a convolutional layer that takes as input the noise residuals generated from the HPF layer. This convolutional layer explores relations of neighboring pixels in the residuals and generates feature maps for statistical modeling. To assist the statistical modeling of the CNN, we disable the default bias learning in this convolutional layer so that the feature maps are symmetric with respect to zeros (sign symmetry [3]); then, we insert an ABS layer right after this convolutional layer to discard the signs of the elements in the feature maps. The output of the ABS layer is thereafter fed into a BN layer. The BN layer first normalizes elements in each input feature map to zero mean and unit variance to ensure that the initial input to *TanH* falls in the quasi-linear region, shown in Fig. 3(a), and hence the gradient back-propagation would not fall into poor local minima. Data in the convolutional module of the CNN have four dimensions: (n, k, i, j), n denotes the nth input data in the minibatch in the training stage ($1 \le n \le N$), k denotes the kth feature map ($1 \le k \le K$), i and j ($1 \le i \le H, 1 \le j \le W$) are the spatial height and width indices in the feature maps. Let $x_{n,i,j}^{(k)}$ and $\hat{x}_{n,i,j}^{(k)}$ represent, respectively, the input elements of the kth feature map and the corresponding normalized feature map, and the normalization step for the kth feature maps can be written as

$$\hat{x}_{n,i,j}^{(k)} = \frac{x_{n,i,j}^{(k)} - \mu^{(k)}}{\sigma^{(k)}}, \tag{1}$$

where

$$\mu^{(k)} = \frac{1}{NHW} \sum_{n,i,j} x_{n,i,j}^{(k)},$$

$$\sigma^{(k)} = \sqrt{\frac{1}{NHW} \sum_{n,i,j} \left(x_{n,i,j}^{(k)} - \mu^{(k)} \right)^2}.$$

To recover the power of modeling, the BN layer then scales and shifts the normalized data with a scaling factor $\gamma^{(k)}$ and a bias $\beta^{(k)}$, both optimized, for each normalized feature map, and generates an output $y_{n,i,j}^{(k)}$ which can be calculated by

$$y_{n,i,j}^{(k)} = \gamma^{(k)} \hat{x}_{n,i,j}^{(k)} + \beta^{(k)}. \tag{2}$$

As all the normalization, scaling and shifting processing in the BN layer are identical within each feature map [demonstrated in (1)], the spatial correlations between elements are

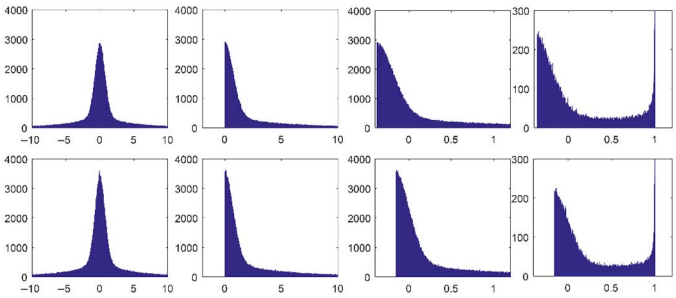

Fig. 2. Distributions of the elements in the first feature map (first row) and those in the second feature map (second row) after going through (a) convolutional layer; (b) ABS layer; (c) BN layer; and (d) TanH layer, in Group 1.

TABLE I
OPTIMIZED SCALES AND BIASES (2) IN THE BN LAYER IN GROUP 1 AFTER THE NORMALIZATION STEP (1)

k	1	2	3	4	5	6	7	8
γ^k	1.20	1.24	1.15	1.61	0.91	0.93	1.04	1.10
β^k	0.29	0.51	0.50	0.55	0.25	0.33	0.58	0.40

well preserved; contrastively, as those processing differ across feature maps, optimal scaling and shifting parameters could be learned for each feature map. Because of the bias terms introduced in the BN layer, without the ABS layer, the sign symmetry of elements in the output feature maps of the first convolutional layer would no longer hold, causing interference between feature values at early stages of statistical modeling. This problem could be solved once the sign information is discarded by the ABS layer.

To provide some intuitive understanding of the functionality in Group 1, we drawn in Fig. 2 the distributions of a test image after it went through the convolutional layer, ABS layer, BN layer, and *TanH* activation in Group 1 of a trained CNN (only the first two feature maps are displayed). Since we disabled bias learning in the first convolutional layer, the output distributions, in Fig. 2(a), are symmetric with respect to zeros. The output of the ABS layer, with the distributions shown in Fig. 2(b), is first normalized following (1) with the global statistics stored during training instead of statistics obtained from the input minibatch. The normalized feature maps are then scaled and shifted (with biases) for optimal statistical modeling; the distributions after this step are displayed in Fig. 2(c). Table I records the optimized scaling and shifting values for all the eight feature maps in the first BN layer. The outputs of the BN layer are then mapped and bounded by the *TanH* activation, as shown in Fig. 2(d).

C. Other Design Considerations

In this section, we describe our considerations on two other parts of the CNN design: the nonlinear activations and spatial sizes of convolutional kernels.

In our CNN designed for steganalysis, a hybrid of *TanH* and *ReLU* nonlinear activations are employed. We discovered that using the *TanH* instead of the more popularly used *ReLU* as nonlinear activations in Group 1 and Group 2 improves performance. This could be attributed to the saturation regions of the *TanH*, which effectively limit the range of data values [illustrated in Figs. 3(a) and 2(d)] and prevent the deeper

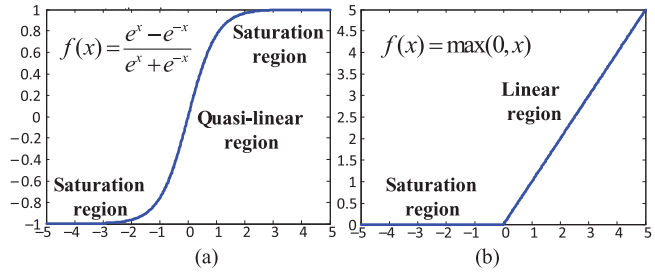

Fig. 3. Two nonlinear activation functions. (a) *TanH*. (b) *ReLU*.

layers from modeling large values, as they are usually sparse and statistically not significant. It is also discovered that results became worse when more *ReLU*s in deeper layers were replaced by the *TanH*, likely due to the difficulty of gradient back-propagation with *TanH* [27].

Conventional machine learning-based steganalysis models patterns of correlations in a small local region of residual elements and adopts one-shot histogram pooling to prevent modeling larger regions [1]–[6]. In contrast, the CNN works by modeling (with alternating convolutional and pooling layers) relationships of residual elements over the whole images. On the one hand, such type of modeling is the root of the strength of the CNN, on the other hand, without effective control, overfitting (learning more image content than stego noise, and/or fitting the stego noise in training data too well to generalize to testing data) could occur for steganalysis. Having realized this potential issue, in the proposed CNN, we limit the sizes of convolutional kernels in deeper layers, i.e., the spatial sizes of the convolutional layers in the last three groups are limited to 1×1 so as to constrain the strength of modeling. Note that the selection of 1×1 convolution should not be taken for granted. It is expected that with more training data, larger convolutional sizes would be preferred.

III. EXPERIMENTS

A. Dataset and Software Platforms

We performed experiments using our designed CNN to detect two spatial domain content-adaptive steganographic algorithms: S-UNIWARD [9] and HILL [11], with embedding rates of 0.1 and 0.4 bpp. The corresponding performance achieved by the SRM [3] with ensemble classifier [25] is used as references. All of the experiments using the CNN were performed on a modified version of Caffe toolbox [28].

The dataset used is BOSSbase v1.01 [22] 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. To avoid confusion during performance comparison in the future works, we used the original image size (512×512) in all the experiments.

B. Training, Validation, and Testing

For each experiment (a reported accuracy in Table II), out of the 10 000 pairs of images, 5000 pairs were set aside for testing

TABLE II
ACCURACY (%) OF CNN AND SRM AGAINST S-UNIWARD AND HILL

	0.1 bpp		0.4 bpp	
	CNN	SRM	CNN	SRM
S-UNIWARD	57.33	59.25	80.24	79.53
HILL	58.44	56.44	79.24	75.47

to verify the performance. These 5000 testing pairs were not touched in the whole training phase.

In the training phase, we obtain totally five optimized CNN models from the 5000 training pairs. Each of the CNN was trained on 4000 pairs and validated on 1000 pairs. The validation pairs were used to decide when to stop training of each CNN. Specifically, we generated five 4000/1000 splits from the 5000 training pairs by evenly breaking the 5000 training pairs into five nonoverlapping folds, and each time used a different fold for validation and the rest four folds for training. Note that this is not cross-validation, it is more like a controlled bootstrap aggregation [30] to create ensemble.

In the testing stage, the 10 000 testing images (5000 testing pairs) went through all the five trained CNNs one by one, and the five output probabilities were averaged for each test image to make the final prediction.

C. Hyperparameters

Minibatch gradient descent was used to train all the CNNs in experiments. The momentum was fixed to 0.9. The learning rate was initialized to 0.001 and scheduled to decrease 10% for every 5000 iterations, for all the parameters. A minibatch of 64 images (32 cover/stego pairs) was input for each iteration. All of the CNNs were trained for 120 000 iterations. Parameters in convolution kernels were initialized by random numbers generated from zero-mean Gaussian distribution with standard deviation of 0.01; bias learning was disabled in convolutional layers and fullfilled in BN layers. Parameters in the last FC layers were initialized using "Xavier" initialization [27]. Except for the FC layer, weight decay (L2 regularization) was not enabled so that we could focus on designing the CNN architecture. For the same reason, no "dropout" [29] was used.

D. Results

The experiments using the SRM (with ensemble classifiers [25]) were conducted on the same 5000/5000 train/test split as for the CNNs. To evaluate the performance, we used the average accuracies, recorded in Table II and the receiver operating characteristic (ROC) curves together with the corresponding area under ROC curves (AUC), illustrated in Fig. 4. Overall, the CNN has competitive performance compared with the SRM and ensemble classifier. Table III reports the means and standard deviations of the testing accuracy obtained from the five single CNN models.

In Table IV, we record the losses in validation accuracies by adjusting the strutures of the CNN to support our design in Sections II-B and II-C. The baseline CNN is one of the single models in Table III to detect S-UNIWARD at 0.4 bpp. These results verified the effectiveness of our proposed design.

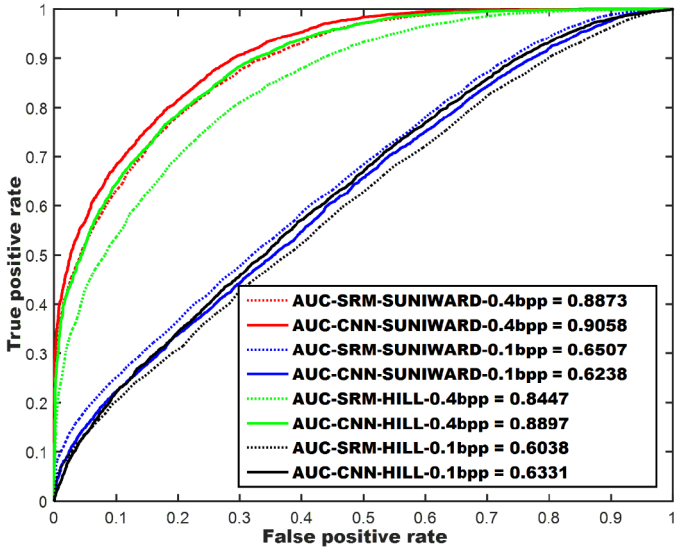

Fig. 4. ROC curves and their corresponding AUC illustrating the overall performance of CNN and SRM on detecting S-UNIWARD and HILL.

TABLE III
MEANS AND STD OF SINGLE CNN MODEL ACCURACIES (%)

	SUNIWARD		HILL	
	0.1 bpp	0.4 bpp	0.1 bpp	0.4 bpp
mean	56.85	79.03	57.55	77.58
std	0.91	0.67	0.68	0.30

TABLE IV
LOSS IN ACCURACY (%) WITH DIFFERENT CNN STRUCTURES

3×3	5×5	wo-abs	rp-tanh
0.47	1.34	3.30	2.24

3×3 = replace all 1×1 convolution kernels with 3×3 kernels.
5×5 = replace all 1×1 convolution kernels with 5×5 kernels.
wo-abs = without the absolute layer.
rp-tanh = replace the *TanH* layer in Group 2 with *ReLU*.

In particular, adding the absolute layer to enhance statistical modeling and using the *TanH* activation after the first two convolutional layers have resulted in large improvements on the detection performance.

IV. CONCLUSION

In this letter, it is shown that a well-designed CNN is a good steganalytic tool and would have the potential, in the future, to provide a better detection performance compared with that achieved by the conventional feature-based steganalysis. Currently, the performance of the proposed CNN still could not compare with some more advanced steganalytic methods [31]–[34], which rely on knowing the probability maps generated during data embedding. It would be interesting future works to feed the CNN with, e.g., the probability maps of embedding, to further enhance the performance against content-adaptive steganography. Also, how to apply the CNN in the best way to defeat steganography in the JPEG domain [35]–[37] would be another important future work.

REFERENCES

[1] D. Zou, Y. Q. Shi, W. Su, and G. Xuan, "Steganalysis based on Markov Model of thresholded prediction-error image," in *Proc. IEEE Int. Conf. Multimedia Expo*, Toronto, ON, Canada, Jul. 9–12, 2006, pp. 1365–1368.

[2] Y. Q. Shi, P. Sutthiwan, and L. Chen, "Textural features for steganalysis," in *Proc. 14th Int. Conf. Inf. Hiding*, Berkeley, CA, USA, May 2012, vol. 7692, pp. 63–77.

[3] J. Fridrich and J. Kodovský, "Rich models for steganalysis of digital images," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 3, pp. 868–882, Jun. 2012.

[4] T. Pevný, P. Bas, and J. Fridrich, "Steganalysis by subtractive pixel adjacency matrix," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 2, pp. 215–224, Jun. 2010.

[5] V. Holub and J. Fridrich, "Random projections of residuals for digital image steganalysis," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 12, pp. 1996–2006, Dec. 2013.

[6] L. Chen, Y. Q. Shi, and P. Sutthiwan, "Variable multi-dimensional co-occurrence histogram for steganalysis," in *Proc. IWDW*, Oct. 2014, vol. 9023, pp. 559–573.

[7] T. Pevný, T. Filler, and P. Bas, "Using high-dimensional image models to perform highly undetectable steganography," in *Proc. 12th Int. Workshop Inf. Hiding*, New York, NY, USA: Springer-Verlag, Jun. 28–30, 2010, vol. 6387, pp. 161–177.

[8] V. Holub and J. Fridrich, "Designing steganographic distortion using directional filters," in *Proc. 4th IEEE Int. Workshop Inf. Forensics Security*, Dec. 2012, pp. 234–239.

[9] V. Holub, J. Fridrich, and T. Denemark, "Universal distortion function for steganography in an arbitrary domain," *EURASIP J. Inf. Secur.*, vol. 2014, no. 1, pp. 1–13, 2014.

[10] V. Sedighi, R. Cogranne, and J. Fridrich, "Content-adaptive steganography by minimizing statistical detectability," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 2, pp. 221–234, Feb. 2016.

[11] B. Li, M. Wang, J. Huang, and X. Li, "A new cost function for spatial image steganography," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Paris, France, Oct. 2014, pp. 4206–4210.

[12] B. Li, S. Tan, M. Wang, and J. Huang, "Investigation on cost assignment in spatial image steganography," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 8, pp. 1264–1277, Aug. 2014.

[13] Y. Le Cun *et al.*, "Handwritten digit recognition with a backpropagation neural network," in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed. San Mateo, CA, USA: Morgan Kaufmann, 1990, pp. 396–404.

[14] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.

[15] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, pp. 211–252, 2015.

[16] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recog.*, Jun. 2015, pp. 1–9.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, arXiv: 1502.03167.

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

[19] J. Chen, X. Kang, Y. Liu, and Z. J. Wang, "Median filtering forensics based on convolutional neural networks," *IEEE Signal Process. Lett.*, vol. 22, no. 11, pp. 1849–1853, Nov. 2015.

[20] S. Tan and B. Li, "Stacked convolutional auto-encoders for steganalysis of digital images," in *Proc. Asia Pac. Signal Inf. Process. Assoc. (APSIPA) Annu. Summit Conf.*, Dec. 2014, pp. 1–4.

[21] Y. Qian, J. Dong, W. Wang, and T. Tan, "Deep learning for steganalysis via convolutional neural networks," in *Proc. SPIE Media Watermarking Secur. Forensics*, 2015, vol. 9409, pp. 94090J–94090J-10.

[22] P. Bas, T. Filler, and T. Pevný, "Break our steganographic system—The ins and outs of organizing BOSS," in *Proc. Inf. Hiding*, May 2011, vol. 6958, pp. 59–70.

[23] A. D. Ker, R. Böhme, E. J. Delp, and P. W. Wong, Eds., "Revisiting weighted stego-image steganalysis," in *Proc. SPIE, Electron. Imag. Secur. Forensics Steganogr. Watermark. Multimedia Contents X*, San Jose, CA, USA, Jan. 27–31, 2008, vol. 6819, pp. 5 1–5 17.

[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

[25] J. Kodovsky, J. Fridrich, and V. Holub, "Ensemble classifiers for steganalysis of digital media," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 432–444, Apr. 2012.

[26] 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 source mismatch," in *Proc. SPIE*, Feb. 2016, pp. 14–18.

[27] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Artif. Intell. Statist. (AISTATS)*, 2010, pp. 249–256.

[28] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[29] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[30] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[31] T. Denemark, V. Sedighi, V. Holub, R. Cogranne, and J. Fridrich, "Selection-channel-aware rich model for steganalysis of digital images," in *Proc. IEEE Int. Workshop Inf. Forensics Security (WIFS)*, 2014, pp. 48–53.

[32] W. Tang, H. Li, W. Luo, and J. Huang, "Adaptive steganalysis against WOW embedding algorithm," in *Proc. ACM Workshop Inf. Hiding Multimedia Secur. (IH&MMSec)*, Jun. 2014, pp. 91–96.

[33] W. Tang, H. Li, W. Luo, and J. Huang, "Adaptive steganalysis based on embedding probabilities of pixels," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 734–745, Apr. 2016.

[34] T. Denemark, J. Fridrich, and P. Comesaña-Alfaro, "Improving selection-channel-aware steganalysis features," in *Proc. SPIE*, Feb. 2016, pp. 14–18.

[35] V. Holub and J. Fridrich, "Low-complexity features for JPEG steganalysis using undecimated DCT," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 2, pp. 219–228, Feb. 2015.

[36] J. Kodovsky and J. Fridrich, "Steganalysis of JPEG images using rich models," in *Proc. SPIE Media Watermark. Secur. Forensics*, Feb. 2012, vol. 8303, Art. no. 83030A-13.

[37] X. Song, F. Liu, C. Yang, X. Luo, and Y. Zhang, "Steganalysis of adaptive JPEG steganography using 2D Gabor filters," in *Proc. ACM Workshop Inf. Hiding Multimedia Secur. (IH&MMSec)*, 2015, pp. 15–23.