Stacked Convolutional Auto-Encoders for Steganalysis of Digital Images

Shunquan Tan* and Bin Li†
*Shenzhen Key Laboratory of Media Security,
College of Computer Science & Software Engineering, Shenzhen University, Guangdong Province, 518060 China
E-mail: tansq@szu.edu.cn
†Shenzhen Key Laboratory of Media Security,
College of Information Engineering, Shenzhen University, Guangdong Province, 518060 China
E-mail: libin@szu.edu.cn

***Abstract*—In this paper, we point out that SRM (Spatial-domain Rich Model), the most successful steganalysis framework of digital images possesses a similar architecture to CNN (convolutional neural network). The reasonable expectation is that the steganalysis performance of a well-trained CNN should be comparable to or even better than that of the hand-coded SRM. However, a CNN without pre-training always get stuck at local plateaus or even diverge which result in rather poor solutions. In order to circumvent this obstacle, we use convolutional auto-encoder in the pre-training procedure. A stack of convolutional auto-encoders forms a CNN. The experimental results show that initializing a CNN with the mixture of the filters from a trained stack of convolutional auto-encoders and feature pooling layers, although still can not compete with SRM, yields superior performance compared to traditional CNN for the detection of HUGO generated stego images in BOSSBase image database.**

I. Introduction

Image steganalysis is the art of detecting data hidden in cover images by means of steganography. In order to improve security, modern content-adaptive steganography constrains embedding changes to edge and texture regions where the statistics are hard to model in practice. Advanced content-adaptive steganographic methods, such as HUGO [1], pose great challenge to steganalyzers. SPAM [2], the once most successful feature-based blind steganalytic method, shows poor performance when attacking HUGO [1]. As far as we know, the current state-of-the-art feature-based blind steganalyzers which can reliably detect stego images generated by content-adaptive steganographic methods are SRM and its descendants [3], [4], [5].

SRM, the Spatial-domain Rich Model, adopts a 34,671-dimensional hardwired image feature descriptor and uses the ensemble classifier, a random forest consisting of multiple FLD (Fisher Linear Discriminants) based minor learners as described in [6] to learn a binary steganalyzer. However, theoretical analysis reveals that ensembles of simple base learners being more powerful than a single base learner is mainly because of the extra level they add to the learning architecture [7], which raises a question to researchers: will the introduction of more deeper learning layers be beneficial to feature-based blind steganalysis? In fact, deep learning architectures, including DBN (Deep Belief Network), SAE (Stacked Auto-Encoders), CNN (Convolutional Neural Networks) and their variants have been applied with great success in various scientific areas [7]. But to the best knowledge of the authors of this paper, there has still been no reports on applications of deep learning architectures in steganalysis. In this paper, we point out that SRM possesses a similar architecture to CNN [8], one of the major deep learning architectures. A nine-layer, three-stage CNN based blind steganalyzer is constructed which accepts raw image pixels as its input and outputs the binary classification results which can be used to distinguish stego images from cover images. CAE (Convolutional Auto-Encoder) are used in a layer-wise unsupervised pre-training procedure. CNN is initialized from the resulting SCAE (Stack of CAEs) [9] with identical topology. The experimental results show that the trained CNN (SCAE) based steganalyzer, although still can not compete with SRM, yields superior performance compared to traditional CNN for the detection of HUGO generated stego images in BOSSBase image database.

The paper is organized as follows. Section II gives a brief overview of SRM. Section III shows the constructional similarity between SRM and CNN firstly, and then reveals the structure of the specific CNN steganalyzer and the details of the unsupervised pre-training procedure using SCAE. Section IV presents experimental results. Finally, concluding remarks are given in Section V.

II. Overview of SRM

The core idea of SRM is to capture diverse types of dependencies among neighboring pixels, which guarantees the superior adaptability of the resulting model [3]. The dependencies are modeled as noise residuals. Let X_{ij} be the pixel located at (i,j) of the target image $\mathbf{X}$, $\mathcal{N}_{ij}$ be a local neighborhood of pixel X_{ij}, $\theta(\mathcal{N}_{ij})$ be a predictor of X_{ij} defined on $\mathcal{N}_{ij}$, the noise residuals, $\mathbf{R} = (R_{ij}) \in \mathbb{R}^{n_1 \times n_2}$, are computed using the following form:

$$R_{ij} = \theta(\mathcal{N}_{ij}) - X_{ij} \tag{1}$$

This work was supported by National Natural Science Foundation of China (61402295, 61103174), Foundation for Distinguished Young Talents in Higher Education of Guangdong (2012LYM_0117) and Fundamental Research Program of Shenzhen City (JCYJ20120613113535357).

978-616-361-823-8 © 2014 APSIPA

All of the pixel predictors are implemented as locally-supported linear filters and can be expressed as the convolutions of $\mathbf{X}$ and a kernel matrix. A total of 39 kernels are used in SRM. For example, the two kernels

$$\mathbf{K}_3 = \frac{1}{4}\begin{pmatrix} -1 & 2 & -1 \\ 2 & 0 & 2 \\ -1 & 2 & -1 \end{pmatrix} \tag{2}$$

$$\mathbf{K}_5 = \frac{1}{12}\begin{pmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & 0 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{pmatrix} \tag{3}$$

predict the value of the central pixel from its local 3×3 and 5×5 neighborhoods, respectively. Other kernels involve pixels arranged only in a horizontal/vertical direction which derived from constant, linear and quadratic models of local image patches, or use special parts of K_3 and K_5 to provide better estimations at spatial discontinuities. Two or more residuals can be merged to form a "minmax" type residual by taking the minimum (or maximum) of the filters' outputs.

Different quantized and truncated versions of each residual are calculated:

$$R_{ij} \leftarrow \mathrm{trunc}_T \left(\mathrm{round} \left(\frac{R_{ij}}{q} \right) \right) \tag{4}$$

where T is a truncation threshold and q is a quantization step. Having obtained the quantized residuals, the SRM submodels will be constructed from their horizontal and vertical $4th$ order co-occurrences. The $4th$ order horizontal co-occurrence matrix of residual (R_{ij}) is defined as the normalized number of the groups of four neighboring residual samples with values equal to d_1, d_2, d_3, d_4:

$$\mathbf{C}^{(h)}_{d_1,d_2,d_3,d_4} = \frac{1}{Z} \big| \{ (R_{ij}, R_{i,j+1}, R_{i,j+2}, R_{i,j+3}) | \\ R_{i,j+k-1} = d_k, k = 1, \ldots, 4 \} \big| \tag{5}$$

The corresponding vertical co-occurrence matrix is defined analogically. Some post procedures are followed by leveraging symmetries of natural images. When all the resulting submodels are put together, their combined dimensionality is $34,671$.

III. SCAE for Steganalysis

SRM can be regarded as a single-stage feature extraction system. Its first layer is a filter bank hardwired with diverse types of edge detectors. The quantization and truncation in (4) is the second layer which bring in nonlinearities. The third layer is to obtain the $4th$ order horizontal and vertical co-occurrence matrices (See (5)) of the resulting residuals, which is indeed a pooling operation that combines nearby values in residual space through histogramming operator. The ensemble classifier which trained in purely supervised mode is the last and the top layer.

By virtue of the structure, SRM exhibits similarity to CNN. In one stage of CNN, the first layer is also an array of band-pass filters, which followed by point-wise sigmoid nonlinearities. The second layer is the average/max pooling and subsampling layer. This layer is used in CNN to reduce the dimensionality, and plays the same role as quantization and truncation in SRM. A complete CNN are composed of several such stages, which assemble in order and followed by a classifier. Several sequences of CNN stages acts similarly to the pooling operation provided by co-occurrences in SRM. The classifier employed by CNN can be a simple linear classifier, a fully connected neural network, or even ensemble classifier, like the one SRM adopted. The primary difference between SRM and CNN is that the first layer of SRM is hardwired with diverse types of edge detectors, while in CNN the filter banks are initialized with random values and is trained in supervised mode.

The authors of SRM attribute the success of the model to the special structure of the filters and their diversity. One reasonable expectation is that a CNN with similar structure to SRM can learn its own filter kernels from samples and gain the same capacity. Furthermore, former researches in this field reveal that the feature extraction systems with two or more stages are systematically and significantly better than their single-stage counterparts [10]. However, it is quite time intensive to train deep learner to yield state-of-the-art results. Hence we did an overall consideration of the complexity of the problem and the ability of the hardwares we equipped with and put forward a nine-layer, three-stage CNN, which is illustrated in Fig.1. The first stage of the proposed CNN takes 512×512 target image as input. It consists of a convolutional layer with forty 5×5 kernels and a max-pooling layer with 4×4 downsampling. The forty kernels used in the convolutional layer resembles the filter bank in SRM. The output of the first stage is forty 128×128 features map, which acts as the input of the consequently second stage. The second stage equipped with a convolutional layer with ten 5×5 kernels and a max-pooling layer with 4×4 downsampling. Its output is a set of total four hundred 32×32 features maps. The third stage has the same structure as the second one, which takes the output of the second stage as its input and generates four thousand 8×8 feature maps. A fully connected neural network is adopted by the proposed CNN based steganalyzer as binary classifier. The output values of the four thousand 8×8 features maps from the third stage are concatenated together in a zigzag order to form the $25,600$ dimensional input of the neural network. The neural network contains one hidden layer with $3,000$ hidden neurons and an output layer with two neurons, in which one stands for "stego" classification result and another stands for "cover".

However, many researches [7] reported that when starting from random initialization, the training procedure of deep multi-layer neural networks including CNN tend to stuck at local plateaus or even diverge which result in rather poor solutions. This phenomenon is also verified by our extensive experiments. In order to circumvent this obstacle, firstly we multiply the kernels of the first convolutional layer by K_5 (Eq. (3)) after initializing them with evenly distributed random values, with the intention of endowing the initial

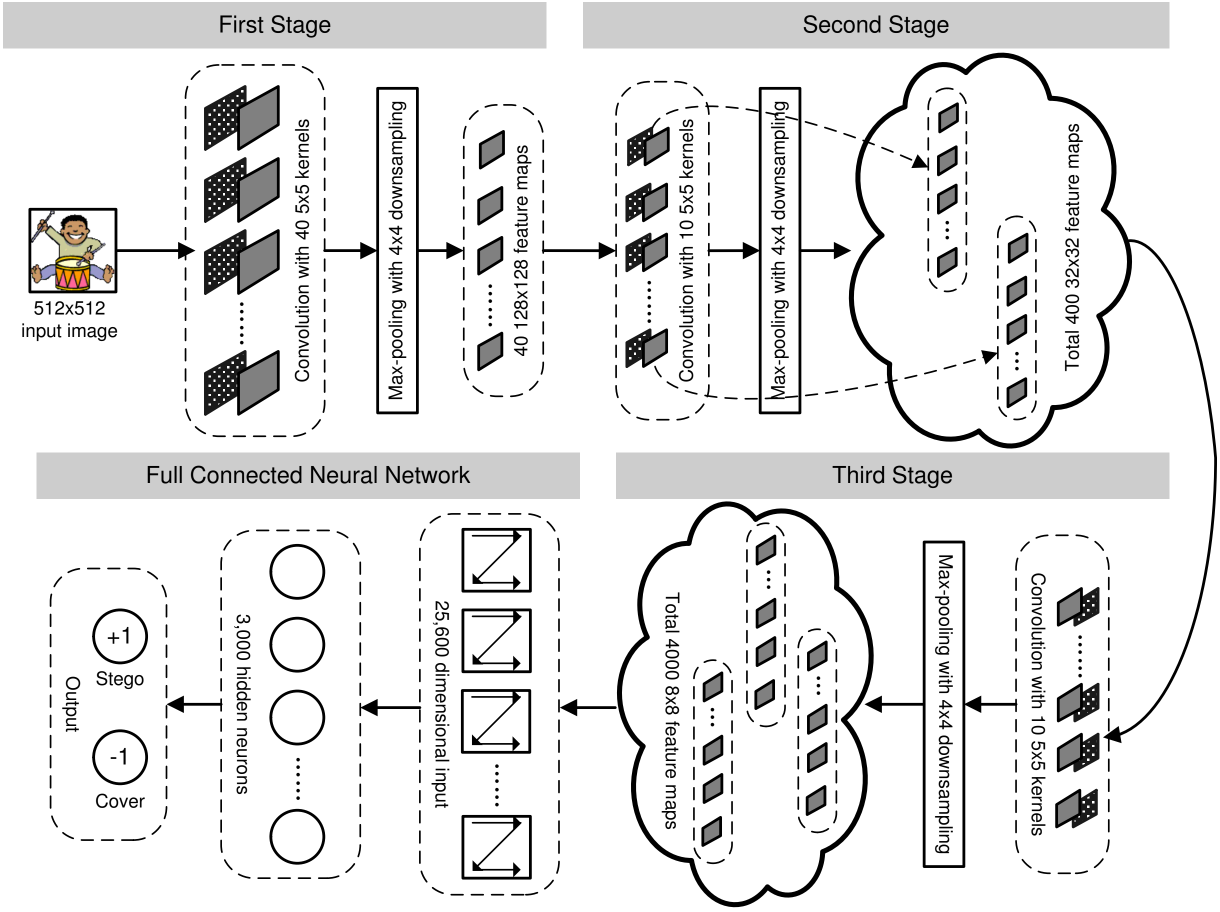
First Stage
Second Stage
512x512
input image
Convolution with 40 5x5 kernels
Max-pooling with 4x4 downsampling
40 128x128 feature maps
Convolution with 10 5x5 kernels
Max-pooling with 4x4 downsampling
Total 400 32x32 feature maps
Full Connected Neural Network
Third Stage
Output
+1
Stego
-1
Cover
3,000 hidden neurons
25,600 dimensional input
Total 4000 8x8 feature maps
Max-pooling with 4x4 downsampling
Convolution with 10 5x5 kernels

Fig. 1. The proposed nine-layer, three-stage CNN (SCAE) based blind steganalyzer.

state of the convolution kernels in the first stage with edge detector-liked structure. The logic is that, since the success of SRM owes to special structure of the filters, initializing the corresponding convolution kernels of our model to a similar structure may aid in the convergence of the model.

The second measure is the introduction of SCAE [9] based unsupervised pre-training procedure. SCAE (Stacked Convolutional Auto-Encoders) are, as the name suggests, convolutional auto-encoders stacked on top of each other, and trained in a layerwise greedy fashion. Its building brick, a CAE (Convolutional Auto-Encoder) is a discriminative graphical model that takes feature maps as input and attempts to reconstruct them via minimizing an appropriate cost function over the training samples. An illustration of the CAE used in the unsupervised pre-training procedure is shown in Fig. 2. One CAE is constructed for each stage of the proposed CNN (SCAE) based blind steganalyzer. From Fig. 2 we can see that the left side of the CAE is of identical topology and initial parameters (especially the special initial state of the kernels, as the last paragraphs mentioned) as the CNN stage it corresponding to. The input feature maps are convolved with the input kernels in the convolution layer and then pass through the max-pooling layer. The intermediate feature maps, the output of the left side of CAE then continue to pass through the 4×4 upsampling layer, and the output convolutional layer which contains the same number kernels as its counterpart, the input convolutional layer of the left side. The output of the right side are the estimations of the initial input feature maps. The unsupervised pre-training procedure minimize the cost function, in this paper the mean squared error between the input feature maps and the output estimations of CAE. A back-propagation algorithm is applied to tune the parameters of the convolutional kernels. The CAEs of the corresponding three stages can be stacked to form a SCAE. Each layer receives its input from the output of the previous stage in the pre-training procedure. The layers of each CAE which take output of the previous stage and generate the intermediate feature maps (the ones within the dotted-line circle in Fig. 2) are indeed the corresponding CNN stage of the proposed steganalyzer. After pre-training, they are extracted from each CAE of the trained SCAE and reassemble stage by stage to build up the proposed CNN (SCAE) based blind steganalyzer prior to a final supervised training stage. It is believed that [7] unsupervised layer-wise learning can help discovering a representation that captures statistical regularities of each layer's input and move the model parameters in a favorable direction. The benefits of unsupervised SCAE based pre-training are also investigated in our experiments.

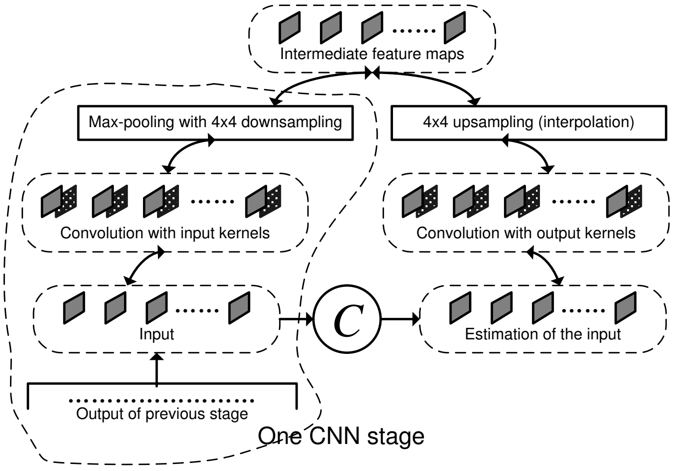
Intermediate feature maps
Max-pooling with 4x4 downsampling
4x4 upsampling (interpolation)
Convolution with input kernels
Convolution with output kernels
Input
C
Estimation of the input
Output of previous stage
One CNN stage

Fig. 2. The CAE used in the pre-training procedure of the proposed CNN (SCAE) based blind steganalyzer. The ring in the center with "C" mark denotes the cost function to be minimized. The layers within the dotted-line circle compose the CNN stage which we can extract from the trained CAE.

IV. Experimental Results

The experiments are performed on the BOSSBase (version 1.01) image database which contains 10,000 512×512 grey-scale cover images[1]. The image database are splitted into 5000 training and 5000 testing images. The corresponding HUGO stego images are generated for a fixed payload of 0.4 bpp with model correction [1]. In the unsupervised pre-training procedure, only the 5000 training cover images are used as the input feature maps of the CAE which corresponding to the first CNN stage. While in the consequent supervised training procedure, All the training cover images and the corresponding stego images are used to fine-tune the proposed CNN (SCAE) based blind steganalyzer. A modified version of DeepLearnToolbox [11] is used in our experiments. The performance of the proposed steganalyzer is evaluated using the detection error on the testing image set:

$$P_E \triangleq \min_{P_{FA}} \frac{1}{2}(P_{FA} + P_{MD}(P_{FA})) \tag{6}$$

where P_{FA} and P_{MD} are the probabilities of false alarm and missed detection. Table I reported the detection error for our CNN (SCAE) based blind steganalyzer and its competitors. "Proposed$_1$" denotes the proposed steganalyzer with random initialization; "Proposed$_2$" denotes Proposed$_1$ coupled with the kernels of the first convolutional layer multiplied by K_5; "Proposed$_3$" denotes Proposed$_2$ plus SCAE pre-training. From Tab. I we can see that it is close to random guessing when using the proposed steganalyzer with random initialization, which means that the CNN based steganalyzer may stuck at local plateaus during the training procedure. Multiplying the kernels of the first convolutional layer can promote the performance of the proposed steganalyzer, which is still merely comparable to what SPAM achieved. The introduction of SCAE based pre-training procedure brings a qualitative leap.

TABLE I
Detection error for the proposed CNN (SCAE) based blind steganalyzer and its competitors.

Proposed$_1$	Proposed$_2$	Proposed$_3$	SPAM	SRM
0.48	0.43	0.31	0.42	0.14

However, $P_E = 0.31$, the best achievement of our steganalyzer still can not take rank with what SRM had scored. The experimental results are merely fair, which may be imputed to two reasons: the inadequate scale of the CNN framework employed and the prohibitively long training time of the model. The final blind steganalyzer took approximately one week to train in an Intel E5 server, which made parameter adjustment and infrastructure optimization much difficult than desired.

V. Concluding Remarks

In this paper we put forward a nine-layer, three-stage CNN (SCAE) based blind steganalyzer. In theory the proposed steganalyzer should exhibit similar or even better performance compared with the well-known SRM. But the experimental results were not encouraging. Even with the help of SCAE pre-training, the proposed steganalyzer is, although better than SPAM, still inferior to SRM. However, it is just the beginning of our adventure in deep learning based steganalysis. We believe that the existing obstacles in experiments will be surmounted via the future introduction of GPU based deep-learning infrastructure [12].

References

[1] T. Pevny, T. Filler, and P. Bas, "Using high-dimensional image models to perform highly undetectable steganography," in *Proc. Information Hiding Workshop (IH'2010)*, pp. 161–177, 2010.

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

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

[4] 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, 2013.

[5] W. Tang, H. Li, W. Luo, and J. Huang, "Adaptive steganalysis against WOW embedding algorithm," in *Proc. 2nd ACM Information Hiding and Multimedia Security Workshop (IH&MMSec' 14)*, pp. 91–96, 2014.

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

[7] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[9] J. Masci, U. Meier, D. Ciresan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Proc. 21th international conference on artificial neural networks (ICANN'11)*, 2011.

[10] K. Jarrett, K. Kavukcuoglu, M.A. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. International Conference on Computer Vision (ICCV'09)*, 2009.

[11] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," M.S. thesis, 2012.

[12] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and A. Ng, "Deep learning with COTS HPC systems", in *Proc. 30th international conference on machine learning (ICML'13)*, 2013.

[1] http://exile.felk.cvut.cz/boss