Deep learning for steganalysis via convolutional neural networks

Yinlong Qian[a,b], Jing Dong[b], Wei Wang[b], and Tieniu Tan[b]

[a]Department of Automation, University of Science and Technology of China;
[b]Center for Research on Intelligent Perception and Computing, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences.

ABSTRACT

Current work on steganalysis for digital images is focused on the construction of complex handcrafted features. This paper proposes a new paradigm for steganalysis to learn features automatically via deep learning models. We novelly propose a customized Convolutional Neural Network for steganalysis. The proposed model can capture the complex dependencies that are useful for steganalysis. Compared with existing schemes, this model can automatically learn feature representations with several convolutional layers. The feature extraction and classification steps are unified under a single architecture, which means the guidance of classification can be used during the feature extraction step. We demonstrate the effectiveness of the proposed model on three state-of-the-art spatial domain steganographic algorithms - HUGO, WOW, and S-UNIWARD. Compared to the Spatial Rich Model (SRM), our model achieves comparable performance on BOSSbase and the realistic and large ImageNet database.

Keywords: Steganalysis, Deep Learning, Feature Learning, Convolutional Neural Networks, Gaussian Non-linearity

1. INTRODUCTION

Steganalysis has been extensively studied in the last decade. Its main purpose is to detect the presence of secret messages in digital covers such as digital images coming from a known source. Usually, this task is formulated as a binary classification problem to distinguish between cover and stego objects. Under this case, existing methods mainly build steganalysis detectors in two steps: feature extraction and classification. In the feature extraction step, a set of handcrafted features is extracted from each image to capture the impact of embedding operations. The success of steganalysis generally depends on the feature design. However, the lack of accurate models of natural images complicates this work. Hence, various heuristic methods are proposed. The most reliable feature design paradigm starts with computing a noise residual and then models the residual using conditional or joint probability distributions of adjacent elements.[1] With the increased sophistication of steganography methods, more complex statistical dependencies among individual elements are considered. In recent years, researchers utilize more kinds of noise residuals to obtain rich image representations.[2–5] To summarize, to obtain a more accurate detection on complex image sources, high-dimensional representation is necessary in modern steganalysis, which makes feature design more and more difficult.

In the classification step, classifiers such as SVM or ensemble classifiers are learned based on the extracted features. Since the feature extraction and classification steps are separated, they cannot be optimized simultaneously. This means the guidance of classification cannot be utilized to capture useful information in the feature extraction step.

All the issues discussed above motivate us to learn feature representations for steganalysis instead of spending time trying to engineer new features by hand. Deep learning models are such a class of machines that can learn feature representations automatically. Inspired by the fact that human brain processes information in hierarchical

Further author information: (Send correspondence to J.D.)
ylqian@mail.ustc.edu.cn
{jdong, wwang, tnt}@nlpr.ia.ac.cn

Media Watermarking, Security, and Forensics 2015, edited by Adnan M. Alattar, Nasir D. Memon, Chad D. Heitzenrater
Proc. of SPIE-IS&T Vol. 9409, 94090J · © 2015 SPIE-IS&T · CCC code: 0277-786X/15/$18 · doi: 10.1117/12.2083479

ways with a deep architecture, researchers had expected for decades to train deep multi-layer neural networks. But no success was reported until 2006, when a breakthrough was initiated by Geoff Hinton.[6] In that work, Hinton introduced Deep Belief Networks (DBNs) with a greedy layerwise unsupervised pre-training procedure that can learn a hierarchy of features one level at a time. Many other deep learning models were proposed with the similar manner, such as Deep Boltzmann Machines,[7] deep autoencoders,[8] and Convolutional Neural Networks (CNNs).[9] Those models have deep architectures that consist of multiple levels of non-linear operations and can be trained using either supervised or unsupervised approaches to learn hierarchical representations by building high-level features from low level ones. Generally, deep architectures are able to represent some functions which are not efficiently representable by shallow architectures. They have practically proved to be more powerful learning schemes for many artificial intelligence (AI) tasks such as object recognition[9–11] and natural language processing.[12]

However the steganalysis task is quite different from these AI ones. In fact, the stego noise to deal with in steganalysis is a kind of very weak signal which usually cannot be perceived by the human perceptual system, and this kind of noise is ignored in AI tasks. Thus, the feature representation in steganalysis should be quite different from that in traditional AI tasks. Actually, we have tested the existing CNN models developed for AI tasks as steganalysis models. But it turns out to be a failure, which means these CNN models are hard to capture the stego signal that is important for steganalysis. Therefore, in the framework of deep learning, we propose a customized CNN model called Gaussian-Neuron CNN (GNCNN) for steganalysis purpose. The primary contributions are outlined as below.

1. This paper introduces deep learning for steganalysis, and we consider feature learning for steganalysis as a brand new paradigm.
2. We propose a customized CNN model called GNCNN which considers some special traces caused by steganography. It and can learn feature representations for steganalysis. Additionally, the guidance of classification can be utilized in the feature extraction step by unifying both steps under a single architecture.
3. We demonstrate that, by using feature representations learned automatically from a deep learning model, we are able to get comparable performance with current state-of-the-art detection methods using sophisticated handcrafted features.

The rest of the paper is organized as follows. After a briefly review of CNNs, in Section 2, we describe the proposed GNCNN model for steganalysis. In Section 3, we present experimental results. Conclusions are drawn in Section 4.

2. THE GNCNN MODEL FOR STEGANALYSIS

As one of the most representative deep learning models, CNNs are hierarchical neural networks in which trainable filters and pooling operations are applied alternatingly to the raw input images, resulting in increasingly hierarchical complex feature representations. They have been applied to objection recognition[11] and image classification tasks,[13] and achieved superior performance.

CNNs combine three architectural ideas as shown in Fig. 1: local regions, shared weights, and pooling.[9] Compared to standard feedforward neural networks with similar size layers, CNNs have much fewer connections and parameters. Hence, they are much easier to train, while theoretically-best performance is likely to be only slightly worse.[13] A typical CNN is composed of two kinds of layers: convolutional layer, which usually follows a pooling operation, and classification layer. They vary in how convolution and pooling operations are realized and how the nets are trained.[14]

In our work, we take CNNs as the basic framework for our steganalysis model for certain considerations. First, CNNs can take raw data as inputs without the need for a feature extraction step. In fact, in CNNs, the feature extraction and classification steps are unified under a single framework. This directly satisfies our purpose that learning feature representations from images instead of treating a CNN as yet another classifier built on existing steganalyis features such as the SRM. Second, CNNs are supervised models. Different from

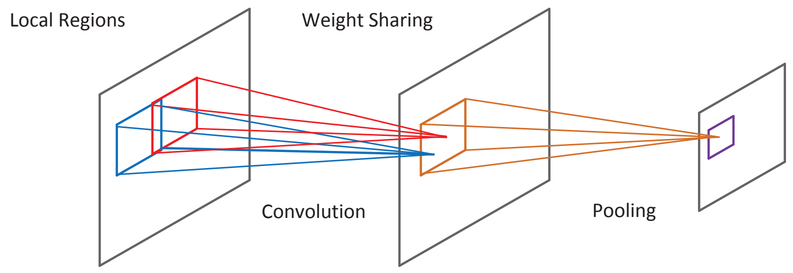

Figure 1. One typical layer of a CNN with convolution and pooling operation. This architecture shows the three basic ideas: local regions, shared weights, and pooling.

some AI tasks, labeled data is quite easy to obtain in steganalysis. Covers and stegos can be regarded as positive and negative samples respectively. In this case, we want to take advantage of such a priori knowledge. Third, with a convolutional learning approach, we are able to train the models on relative large scale images (e.g. 256x256 pixels). In AI tasks, images can be subsampled to small ones (e.g. 32x32 pixels) to make learning computationally tractable. However, this operation erases the stego noise, hence makes detection impossible. So we need a kind of deep learning model that can deal with large scale images.

Though having been successfully applied to some AI tasks, current CNNs do not consider the statistical properties that are important for steganalysis. To effectively capture the useful statistical information for steganalysis, we propose the GNCNN model, a customized deep CNN model. The architecture of our network is summarized in Fig. 2 (right). The left of the figure shows flowchart of traditional steganalysis schemes for comparison. This model takes pixels as inputs, and it is composed of three kind of layers: a image processing layer, several convolutional layers for feature representation, and several fully connected layers for classification. Different from traditional schemes, the parameters are learned automatically.

2.1 Image Processing Layer

The image processing layer is employed to encode our prior knowledge on the design of the network. In this layer, we make a filtering operation with a predefined high-pass filter that are kept fixed during training. Generally, the high frequency stego noise added to the cover is a kind of very weak signal, which is greatly impacted by image content. Hence, with the high pass filtering operation, we aim to strengthen the weak stego signal and reduce the impact of image content. This can provide a good initialization to driven the whole network, hence achieve good performance as compared to random initialization. Actually, this is a commonly used preprocessing skill in traditional steganalysis schemes as mentioned in the introduction. Here we denote **I** as image, **R** as image after high-pass filtering (usually referred to as residual image). Mathematically, **R** can be expressed as below.

$$R = K * I, \tag{1}$$

where the symbol $*$ denotes convolution, and **K** is a shift-invariant finite-impulse response linear filter to compute the residual.

In our framework, instead of the entire family of noise residuals used in current state-of-the-art feature sets such as SRM and PSRM, only one of the relevant filters is used. However, it will be proved that the learned feature representation is comparable to the high dimensional feature sets.

Typically, in our work, the **KV** kernel shown as below is used.

$$K_{kv} = \frac{1}{12}\begin{pmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{pmatrix} \tag{2}$$

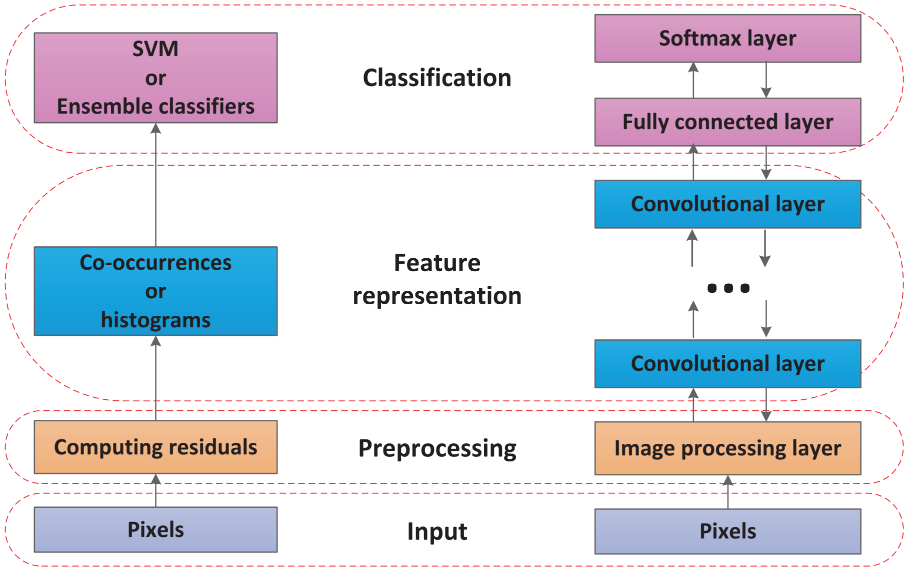

Figure 2. The GNCNN model (right) and traditional steganalysis architecture based on hand-crafted features (left). The up and down arrows in the right flowchart show forward and back propagation directions.

2.2 Convolutional Layer

After applying a preprocessing operation to strengthen the stego signal in the image processing layer, we then hierarchically aggregate the response of stego signal from local to global in the convolutional layers.

The input and output of each convolutional layer are sets of arrays called feature maps. At the output, each feature map is a particular feature representation extracted at all locations on the input. At a convolutional layer, three kinds of operations, which are convolution, non-linearity, and pooling, are usually applied sequentially as expressed below.

$$X_j^l = pool(f(\sum_i X_i^{l-1} * K_{ij}^l + b_j^l)), \tag{3}$$

where $f()$ denotes non-linearity operation, $pool()$ denotes pooling, X_j^l is the j-th feature map in layer l, X_i^{l-1} is the i-th feature map in layer $l-1$, K_{ij} is the trainable convolution kernel connecting the j-th output map and the i-th input map, b_j^l is an trainable bias parameter for the j-th output map.

For the convolution operation, each output feature map usually combines convolutions with multiple input feature maps. The convolutional structure involves the ideas of local regions and shared weights. With local regions, each low level feature will be computed from only a subset of the input such as a neighborhood of a pixel at a given position of an image. Such a local feature extractor shares the same parameters when applied at different neighboring input locations which is equivalent a convolution of the image pixel values with a kernel comprising the weight parameters. The share of parameters produces a shift-invariant operation, and also reduces the number of free variables, hence increases the generalization performance of the network.[15]

In steganalysis, it is considered that, for natural images, subsequent in-camera processing during image acquisition, such as color interpolation, denoising, color correction, and filtering, introduces complex dependencies into noise component of neighboring pixels. But the stego noise caused by steganographic embedding will violate these dependencies. Indeed, most steganalysis methods try to utilize these dependencies to detect the presence of the stego noise. Heuristically, it is supposed that, because of the complex dependencies, a good estimate of the central pixel can be obtained from the neighboring pixels, excluding the pixel being estimated. Then by subtracting the true value of the central pixel from the estimated one, the prediction error value can be obtained, which directly reflects whether the pixel is changed or not. Currently, the successful predictors which

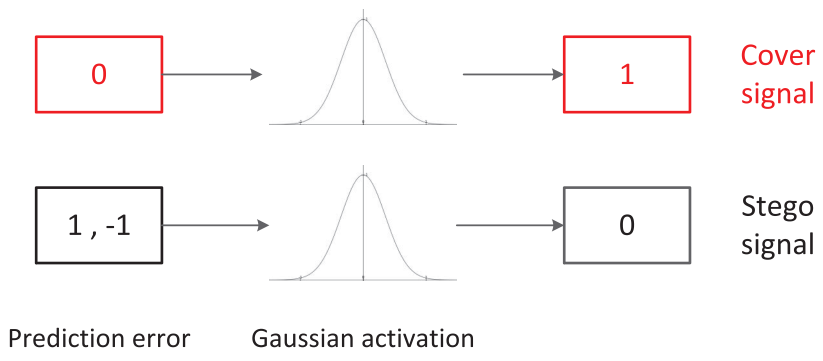

Figure 3. A simple example illustrating why Gaussian activation works for steganalysis in the proposed model. Gaussian activation can distinguish between stego signal and cover signal from the prediction error values.

are usually some shift-invariant filters are hand-designed. In the framework of feature learning, we suppose to learn the predictors automatically. Considering that the filtering operation in the image processing layer has generated a initial noise residual, the convolution operation in each convolutional layer is to hierarchically capture dependencies among a larger neighborhood and to make the prediction more accurate.

Then a non-linear activation function is applied element-wise to the output of convolution operation. The non-linear activation function has the effect of limiting the amplitude of the output. More importantly, for multi-layer networks, it makes possible the solution of some problems that are not possible with linear ones. The choice of activation function is important. It is determined by the nature of the data and the assumed distribution of target variables.[16] Generally, it is chosen to be a sigmoidal function such as the logistic sigmoid or the $tanh()$ sigmoid function in traditional CNNs for AI tasks. But in our work, for steganalysis, we use the Gaussian function, which can be express as

$$f(x) = e^{-\frac{x^2}{\sigma^2}}, \tag{4}$$

where σ is a parameter that determines the width of the curve. A neuron with this activation function will produce a significant positive response only when the input falls into a small interval around zero. Particularly, the maximum response will be obtained at the center zero. To our best knowledge, it is the first time that Gaussian function is used as activation function in deep CNNs, and we call this kind of CNN as Gaussian-Neuron CNN (GNCNN).

As has mentioned, the aim of convolution operation is to compute prediction error values by exploiting the dependencies among neighboring elements. The motivation behind Gaussian non-linearity is that it is better to distinguish between stego signal and cover signal from the prediction error values compared to the sigmoidal one. Fig. 3 gives a simple example of why Gaussian activation works. Ideally, it is supposed that there should be three kinds of prediction error values: 1, -1 and 0. The values 1 and -1 means that the pixel is modified by embedding operation, and we consider there is a stego signal here. The value 0 means that the pixel is unchanged, and we consider there is a cover signal. With Gaussian activation, the prediction error values that corresponding to cover or stego signal will be transformed to 0 or 1 respectively. Hence the cover and stego signal are separated. In actual practice, it is hard to obtain such precise discrete prediction values for every pixel to make a "yes" or "no" decision. Actually, the prediction error values in our framework are continuous, which reflects the strength of stego signal at corresponding positions. With the Gaussian activation function, the values that far away from zero, which are more relevant to stego signal, are transformed to around 0. Meanwhile, the values around zero, which are more relevant to cover signal, are transformed to around 1. Hence, the use of Gaussian activation function provides a novel regularization to enforce the network to learn good predictors for steganalysis.

The resulting activations are then passed to the pooling part of the layer. Pooling operation aims to transform the low level feature representation into a more usable one which preserves important information and discards irrelevant details.[17] In general, higher-level feature representations need information from progressively large

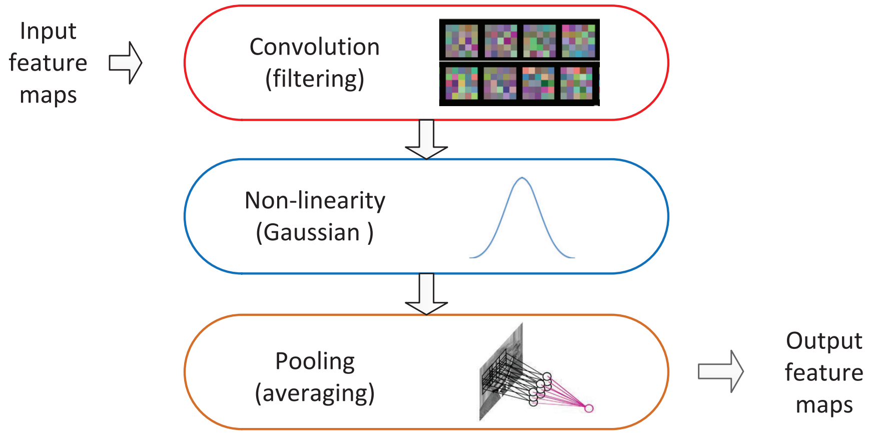

Figure 4. Components of convolutional layer in our model, including convolution, Gaussian non-linearity, and average pooling.

input regions.[18] Pooling has the effect of merging the information within a set of small local regions while reducing computation time. In a pooling operation, the outputs of neighboring groups of neurons in the same feature map are summarized.

Generally, there are two conventional choices for pooling: average pooling and max pooling. The former takes the average value within the pooling region:

$$pool(R_j) = \frac{1}{|R_j|} \sum_{i \in R_j} a_i, \tag{5}$$

while the max pooling operation selects the maximum value:

$$pool(R_j) = \max_{i \in R_j} a_i, \tag{6}$$

where R_j is pooling region j in a feature map, a_i is the i-th element within it.

Max pooling only captures the strongest activation within the pooling region. It is well suited to the the kind of feature representation that are very sparse (i.e. have a very low probability of being active).[17] However for steganalysis, with this kind of operation, some useful information may be lost. The stego signal is a kind of very weak signal, and we consider it not enough to make a accurate prediction when just relying on responses from individual elements. In our proposed model, rather than max pooling, we use average pooling. In average pooling, all the activations in the pooling region are taken into account, which is supposed to discard the distributions caused by individual elements. By merging all the signal within the pooling region, the stego signal on the whole region is strengthen. Actually, in our work, we have tried both pooling operations. The results support our assumption that average pooling has a much better performance than max pooling.

The designed structure in each convolutional layer is summarized in Fig. 4. Each convolutional layer extracts features from all the maps in the previous layer. As the network goes deeper, more complex and higher-level dependencies are modeled by progressively involving large input regions. Hence, the stego signal are aggregated hierarchically from local to global. These high level features learned make it possible for the neurons in the top layer to predict the high-level concept of whether the input region is modified or not.

2.3 Classification Layer

The classification module consists of several fully connected layers. The learned features are passed to these layers. On the top layer, a softmax activation function is used to produce a distribution over all the class labels. In our network, a two-way softmax is used as below.

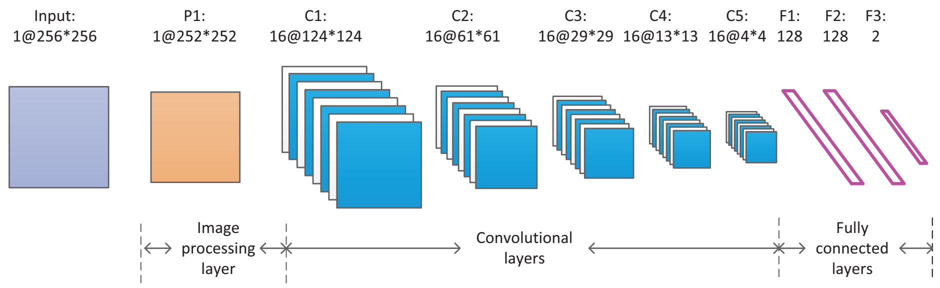

Figure 5. The GNCNN architecture settings in our experiments. This architecture consists of one image processing layer, five convolutional layers, and three fully connected layers. The form "$a@b*b$" means the number of feature maps a and resolution $b*b$ of the corresponding layer. Other detailed descriptions are given in the text.

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{2} e^{x_j}}, \tag{7}$$

for $i = 1, 2$, where x_i is the total input to the neuron i in the top layer, and y_i is its output.

To reduce the problem of overfitting, a recently proposed technique called "dropout" is applied for regularizing fully connected layers.[19] When training with dropout, the output of each neuron in corresponding layers is set to zero with probability 0.5. This technique improves the network's generalization ability and gives an improved test performance.

The GNCNN is trained by minimizing $-logy_t$, where $t \in \{1, 2\}$ denotes the target class, using backpropagation algorithm. Error propagation and weight adaptation in convolutional and fully connected layers follow the standard procedure. Details of the procedure are given in Ref. 9. In a word, all the parameters in feature extraction layers and classification layers are jointly optimized.

3. EXPERIMENTS

In this section, we first describe the settings of the architecture. In this architecture depicted in Fig. 5, we consider an image of size 256×256 as input to the GNCNN model. The first convolutional layer filters the input image with a kernel of size 5×5. The second convolutional layer takes the output of the first layer as input and filters it with 16 kernels of size 5×5. The third, fourth, and fifth convolutional layers apply convolutions with 16 kernels of size 3×3 respectively, while the sixth convolutional layer with 16 kernels of size 5×5. The Gaussian non-linearity is applied to every output of the second to sixth convolutional layers. Meanwhile, each of the second to sixth convolutional layers is followed by an overlapping average pooling operation with window size 3×3 and step size 2, which operates on each of the feature maps in corresponding convolutional layer and leads to the same number of feature maps with reduced spatial resolution.

Finally, the extracted 256 features are passed to the classification module that consists of three fully connected layers. The first two fully connected layers have 128 neurons each. The output of each neuron in the first two fully connected layers of GNCNN is activated by a recent implemented non-saturating non-linearity, the Rectified Linear Units (ReLUs) non-linearity $f(x) = max(0, x)$.[20] CNNs with ReLUs train several times than their equivalents with tanh units.[13] The last fully connected layer has two neurons, and its output is fed to a two-way softmax.

Despite the fact that the proposed GNCNN has much fewer connections and parameters which makes it easier to train, it has still been expensive to apply to high-resolution images. However, recent GPU implementation of convolution has managed to facilitate the training of interestingly large CNNs. All our experiments were conducted using a powerful C++ GPU convolution library,[21] which allowed for rapid experimentation.

In this paper, experiments were carried out on two image sources. The first comes from the standardized database called BOSSbase 1.01.[22] This database contains 10,000 images acquired by seven digital cameras in

Table 1. Detection error of GNCNN model vs. the SRM set implemented with Ensemble classifiers and the SPAM set implemented with a Gaussian SVM for three state-of-the-art spatial domain steganographic algorithms. The dimensionality of corresponding feature set is given in the third row.

bpp	HUGO			WOW			S-UNIWARD		
	GNCNN (256D)	SRM (34,671D)	SPAM (686D)	GNCNN (256D)	SRM (34,671D)	SPAM (686D)	GNCNN (256D)	SRM (34,671D)	SPAM (686D)
					BOSSbase				
0.3	33.8%	29.6%	42.9%	34.3%	31.2%	42.2%	35.9%	31.5%	40.0%
0.4	28.9%	25.2%	39.1%	29.3%	25.7%	38.2%	30.9%	26.3%	35.1%
0.5	25.7%	21.4%	35.7%	24.8%	22.1%	34.9%	26.3%	21.4%	30.6%
					ImageNet				
0.4	33.6%	32.5%	-	34.1%	34.7%	-	34.7%	34.4%	-

Raw format and subsequently processed to the size of 512×512 pixels. In our experiments, we further resized the images to the size of 256×256 pixels. This processing of resizing is only due to the limitation of our computational capabilities. The second image source was obtained from ImageNet, a data set of over 15 million high-resolution images collected from the web. We randomly selected 100,000 images, the shorter side of which is over length 256, from ImageNet. Then we converted them to 8-bit grayscale, followed by resizing so that the smaller side is 256 pixels, and finally cropped out the central 256×256 patches from the resulting images. Compared to the first image source, the second is a highly realistic data set.

To evaluate the effectiveness of the developed model for steganalysis, we ran experiments on three state-of-the-art spatial domain steganographic algorithms: HUGO,[23] WOW,[24] and S-UNIWARD.[25] For the BOSSbase image source, we evaluated the performance of all methods on three payloads: 0.3, 0.4, and 0.5 bits per pixel (bpp). For the larger ImageNet data set, on which the training ran much slower, we evaluated the performance of all methods on payload 0.4 bpp for representation. We picked the trained GNCNN model with the lowest validation error, and evaluated it on the test set. The detection errors are reported in Table 1.

We contrasted the results against two representative handcrafted feature sets: the low-dimensional SPAM set[1] implemented with a Gaussian SVM, and the high-dimensional SRM set,[2] one of the current state-of-the-art feature set, implemented with Ensemble classifiers. All the experiments using traditional handcrafted feature sets for comparison were carried out on the same database we have described before.

Table 1 shows the results of all experiments on both databases. For BOSSbase, across all three embedding algorithms and payloads, our method achieves a much lower detection error than the SPAM set implemented with a Gaussian SVM. When compared to the SRM set implemented with ensemble classifiers, the detection error is just about $2\% - 5\%$ higher depending on the payload. Experiments on ImageNet show that our method achieves a close detection error to the SRM set. Compared to BOSSbase, ImageNet images are collected from the Internet, thus they have much more diversity. However, the data driven deep architecture of our model has the advantage of handling large scale data of great diversity. Since the result on BOSSbase has shown that the detection error for SPAM is much higher than the other two methods and the training time of Gaussian SVM on the large ImageNet would be too long , we did not use SPAM for comparison on ImageNet. All the results of these experiments confirm that the learned feature representations are effective with our model.

In the last part of this section, we discuss the computational cost on training the network. Our network was trained on a PC with Intel Xeon E5-2650 2.0GHz CPU and Tesla K40 12G GPU. The average training time on BOSSbase is about 2 hours. While on the larger ImageNet data set, the average training time is about 52 hours. It is expected that, with multiple GPUs, the training time can be greatly reduced.

4. CONCLUSION

In this paper, we introduce deep learning for steganalysis to learn feature representations automatically. Based on the frame work of CNNs, we have proposed a customized deep model called GNCNN for steganalysis. The proposed model can capture the complex dependencies that are useful for steganalysis, and learn feature representations with several convolutional layers. As the network goes deeper, more complex dependencies are modeled by progressively involving large input regions. Hence, higher level features are obtained. Additionally, both feature extraction and classification are unified under a single network architecture which manages to utilize the guidance of classification by jointly optimizing all the parameters. The final learned representation is a set of 256 features. We have evaluated the performance of our model using BOSSbase and ImageNet data sets. Results show the effectiveness of the GNCNN model on BOSSbase and the realistic and large ImageNet database by achieving a comparable performance compared to the SRM feature set, which demonstrates that the feature representations learned from a deep learning model can achieve competitive performance with handcrafted features. We believe that, by exploring more steganalysis properties, better performance can be achieved.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

REFERENCES

[1] Pevny, T., Bas, P., and Fridrich, J., "Steganalysis by subtractive pixel adjacency matrix," *IEEE Transactions on information Forensics and Security* **5**(2), 215–224 (2010).

[2] Fridrich, J. and Kodovsky, J., "Rich models for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security* **7**(3), 868–882 (2012).

[3] Shi, Y. Q., Sutthiwan, P., and Chen, L., "Textural features for steganalysis," in [*Information Hiding*], 63–77, Springer (2013).

[4] Holub, V., Fridrich, J., and Denemark, T., "Random projections of residuals as an alternative to co-occurrences in steganalysis," in [*IS&T/SPIE Electronic Imaging*], 86650L–86650L, International Society for Optics and Photonics (2013).

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

[6] Hinton, G. E. and Salakhutdinov, R. R., "Reducing the dimensionality of data with neural networks," *Science* **313**(5786), 504–507 (2006).

[7] Salakhutdinov, R. and Hinton, G. E., "Deep boltzmann machines," in [*International Conference on Artificial Intelligence and Statistics*], 448–455 (2009).

[8] Larochelle, H., Bengio, Y., Louradour, J., and Lamblin, P., "Exploring strategies for training deep neural networks," *The Journal of Machine Learning Research* **10**, 1–40 (2009).

[9] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE* **86**(11), 2278–2324 (1998).

[10] Hinton, G., Osindero, S., and Teh, Y.-W., "A fast learning algorithm for deep belief nets," *Neural computation* **18**(7), 1527–1554 (2006).

[11] Ranzato, M., Huang, F. J., Boureau, Y.-L., and LeCun, Y., "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in [*IEEE Conference on Computer Vision and Pattern Recognition*], 1–8, IEEE (2007).

[12] Collobert, R. and Weston, J., "A unified architecture for natural language processing: Deep neural networks with multitask learning," in [*Proceedings of the 25th international conference on Machine learning*], 160–167, ACM (2008).

[13] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet classification with deep convolutional neural networks," in [*Advances in neural information processing systems*], 1097–1105 (2012).

[14] Ciresan, D. C., Meier, U., Masci, J., Maria Gambardella, L., and Schmidhuber, J., "Flexible, high performance convolutional neural networks for image classification," in [*IJCAI Proceedings-International Joint Conference on Artificial Intelligence*], **22**(1), 1237 (2011).

[15] Browne, M. and Ghidary, S. S., "Convolutional neural networks for image processing: an application in robot vision," in [*AI 2003: Advances in Artificial Intelligence*], 641–652, Springer (2003).

[16] Bishop, C. M. et al., [*Pattern recognition and machine learning*], vol. 1, springer New York (2006).

[17] Boureau, Y.-L., Ponce, J., and LeCun, Y., "A theoretical analysis of feature pooling in visual recognition," in [*Proceedings of the 27th International Conference on Machine Learning (ICML-10)*], 111–118 (2010).

[18] Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y., "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in [*Proceedings of the 26th Annual International Conference on Machine Learning*], 609–616, ACM (2009).

[19] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R., "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580* (2012).

[20] Nair, V. and Hinton, G. E., "Rectified linear units improve restricted boltzmann machines," in [*Proceedings of the 27th International Conference on Machine Learning (ICML-10)*], 807–814 (2010).

[21] Krizhevsky, A., "cuda-convnet," (2012). http://code.google.com/p/cuda-convnet/.

[22] Bas, P., Filler, T., and Pevnỳ, T., " break our steganographic system: The ins and outs of organizing boss," in [*Information Hiding*], 59–70, Springer (2011).

[23] Pevnỳ, T., Filler, T., and Bas, P., "Using high-dimensional image models to perform highly undetectable steganography," in [*Information hiding*], 161–177, Springer (2010).

[24] Holub, V. and Fridrich, J., "Designing steganographic distortion using directional filters.," in [*The IEEE International Workshop on Information Forensics and Security (WIFS)*], 234–239 (2012).

[25] Holub, V. and Fridrich, J., "Digital image steganography using universal distortion," in [*Proceedings of the first ACM workshop on Information hiding and multimedia security*], 59–68, ACM (2013).