# An Improved CNN Steganalysis Architecture Based on "Catalyst Kernels" and Transfer Learning

Rabii El Beji, Marwa Saidi(✉), Houcemeddine Hermassi, and Rhouma Rhouma

Ecole Nationale d'Ingénieurs de Tunis, LR16ES07 Robotique, Informatique et Systèmes Complexes, Université de Tunis El Manar, BP. 37 Le Belvédère, 1002 Tunis, Tunisia
marwoua.saidi@gmail.com

**Abstract.** In recent years, the interest of using model architectures based on the Convolutional Neural Networks in steganalysis has been rapidly increasing. Regarding the success of deep learning in the field of imaging analysis. Previous approaches have focused on proposing complex architectures with large sizes of images rather than simple models. In this work, we propose a more adjustable flexible architecture with the use of small size images. The robustness of our method is based on using a set of High Pass Filters (HPF) to extract the residual noise on one hand and exploiting the concept of Transfer Learning ensuring the propagation of optimal weights on the other hand. Three state-of-the-art steganographic algorithms in the spatial domain: WOW, S-UNIWARD, and HUGO are used to evaluate the effectiveness of our classification method. Our proposed technique shows an accelerated convergence for the low payloads and provides a better detection accuracy for the high payloads with a classification accuracy rate crossing 96%.

**Keywords:** Steganalysis · Convolutional Neural Networks
Deep learning · Binary classification · High-Pass Filter (HPF)
Transfer learning

## 1 Introduction

Steganography is the art of hiding secret data through various multimedia supports such as videos, audio files, network packets and last but not least digital images. In the state-of-art literature, various embedding approaches have been proposed, one of the major methods which has been well introduced and used is embedding data through slight alteration of pixels values specifically embedding within the LSBs which referred to the spatial domain embedding. Other methods, in the frequency domain, were based on modifying the DCT coefficients instead of manipulating pixels values which seemed to be more efficient in terms of embedding capacity. The highly undetectable steganographic schemes tend to embed the secret data in the regions with complex content, called as well rich


M. A. Bach Tobji et al. (Eds.): ICDEc 2018, LNBIP 325, pp. 119–128, 2018.
https://doi.org/10.1007/978-3-319-97749-2_9

or textured regions where the embedding artifacts are less detectable. Recently recognized approaches aims to be adaptive to the texture of the image, contrary to previous uniform techniques of embedding. Some of adaptive approaches in the spatial domain, we can include HUGO [1], WOW [2] and S-UNIWARD [3].

As a countermeasure to Steganography, the Steganalysis aims mainly to detect the very existence of potential secret messages statistically within the content of the images and decide a given object is a cover or a stego object. Nowadays, with the enormous evolution of hardware (GPUs) serving to minimize the calculus complexity, techniques of telescopic learning machines such as the Support Vector Machine (SVM), the Ensemble Classifier (EC) or the Deep Learning models such as the Convolutional Neural Networks (CNN or ConvNet) have been introduced and applied within a steganalysis context. The properties of such tools showed an efficient performance in terms of data classification.

Recently, various CNN architectures [5–7,10] have been developed for binary classification. These architectures have been inspired initially from the work of *Qian et al.* [4].

To accelerate the convergence of a given CNN model for a staganographic scheme in the JPEG domain, *Chen et al.* [10] proposed four specific filters fixed in the first layer of their CNN architecture. These filters act as "catalyst kernels" (activator in chemical reactions) for the following layers and accelerate network convergence by ensuring accurate classification between stego and cover images.

Inspired by this latest work, we have exploited the efficiency of these catalyst kernels through a fairly simple though efficient CNN architecture. In this paper, we have proposed a new approach based on the combination of kernels catalyst and transfer learning to ensure rapid convergence for low payloads by weights propagation.

The rest of this paper is organized as follows. In Sect. 2 we outline the details of our proposed method. Experimental results and discussions are presented in Sect. 3. Evaluation results of the proposed architecture are presented in Sect. 4. Conclusion is drawn in Sect. 5.

## 2 Model Description

The methods based on CNN combine the traditional handcraft feature extraction as well as the classification phase. Thus, it improved the classification results through exploiting a back-propagation mechanism.

Typically CNN complex models, which use huge dataset, large image sizes, multi-layers architecture and several neurons, are implemented on a very high-performance GPUs to accelerate network convergence and minimize the training time as much as possible.

Our idea was to build up a model with a minimum calculation cost so that we can do the training on low performing CPUs. To this aim, we reduce the size of the dataset.

Then we outline the design of our CNN network architecture, shown in Fig. 1, with three convolution layers for preprocessing and feature learning and three dense layers for the binary classification.

The preprocessing layer consists of four un-trainable neurons. Then, the second and the third convolutional layers are formed respectively by 5 and 10 neurons. Each neuron has a kernel size of $5 \times 5$ with a stride equal to 2 during the convolution process. A Batch Normalization layer is applied before each non-linear activation function to normalize the weak stego noise and improve the network convergence. The LeaKyReLU function is applied for all activation layers.

Finally, the classification layer consists of 2 fully connected-layer and a softmax layer. Each fully connected layer composed of 200 filters to prepare flatted data to the softmax layer. The softmax layer is a classifier consisting of two kernels, each one gives as an output the probabilities of each class (cover or stego).

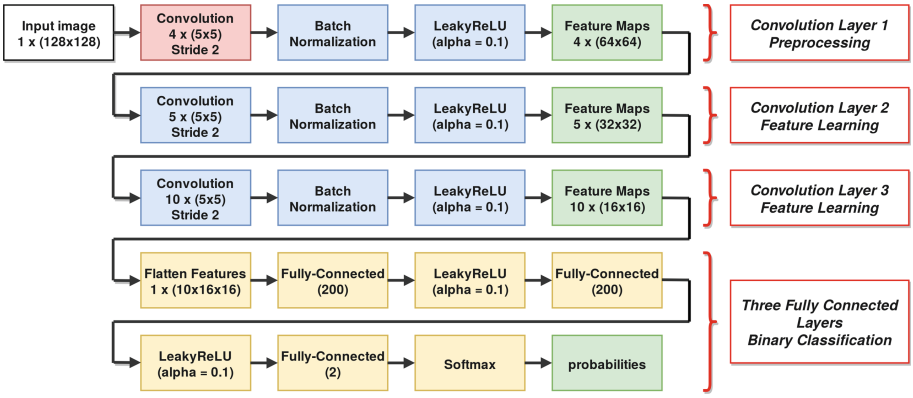



**Fig. 1.** The proposed CNN model architecture.

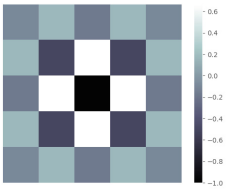
### 2.1 Preprocessing Layer

For the preprocessing layer, we fixed the 4 neurons of the convolution layer with specific filters. These filters shown in Fig. 2 are used together in the work of *Chen et al.* [10], as "catalyst kernels".

The first $5 \times 5$ size $F_{KV}$ filter, was exploited in the Spatial Rich Model (SRM) [8] feature extractor, and later in one of the first CNN steganalysis detectors [4] as well as other models [7,9]. Fundamentally, this filter presented a pre-processing phase of the input data. Due to its sign-changing symmetric checkerboard pattern, It conceal the correlated components while largely preserving the high frequencies of the treated signal.

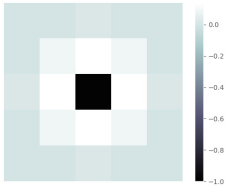
$F_P$ filter is introduced as a high-pass filter selected to complement $F_{KV}$. The $F_H$ and $F_V$ are two second-order horizontal and vertical Gabor filters (1), with $\psi = \pi/2$, $\sigma = 1$, $\theta = 0$ and $\theta = \pi/2$, respectively. These two filters are added within the pre-treatment step due the fact that $F_{KV}$ and $F_P$ are not directional.

$$F_{KV} = \frac{1}{12}\begin{bmatrix} -1 & +2 & -2 & +2 & -1 \\ +2 & -6 & +8 & -6 & +2 \\ -2 & +8 & -12 & +8 & -2 \\ +2 & -6 & +8 & -6 & +2 \\ -1 & +2 & -2 & +2 & -1 \end{bmatrix}$$



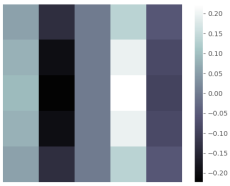
**(a) KV filter and its plot.**

$$F_P = \frac{1}{261}\begin{bmatrix} 0 & 0 & +5.2 & 0 & 0 \\ 0 & +23.4 & +36.4 & +23.4 & 0 \\ +5.2 & +36.4 & -261 & +36.4 & +5.2 \\ 0 & +23.4 & +36.4 & +23.4 & 0 \\ 0 & 0 & +5.2 & 0 & 0 \end{bmatrix}$$



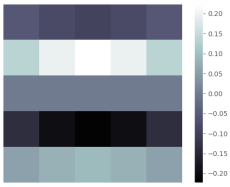
**(b) Point high-pass filter and its plot.**

$$F_H = \begin{bmatrix} +.0562 & -.1354 & 0 & +.1354 & -.0562 \\ +.0818 & -.1970 & 0 & +.1970 & -.0818 \\ +.0926 & -.2233 & 0 & +.2233 & -.0926 \\ +.0818 & -.1970 & 0 & +.1970 & -.0818 \\ +.0562 & -.1354 & 0 & +.1354 & -.0562 \end{bmatrix}$$



**(c)Horizontal 2D Gabor filter and its plot.**

$$F_V = \begin{bmatrix} -.0562 & -.0818 & -.0926 & -.0818 & -.0562 \\ +.1354 & +.1970 & +.2233 & +.1970 & +.1354 \\ 0 & 0 & 0 & 0 & 0 \\ -.1354 & -.1970 & -.2233 & -.1970 & -.1354 \\ +.0562 & +.0818 & +.0926 & +.0818 & +.0562 \end{bmatrix}$$



**(d) Vertical 2D Gabor filter and its plot.**

**Fig. 2.** Four square (5×5) kernels fixed in the first convolutional layer for preprocessing.

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda} + \psi\right) \tag{1}$$

where $x' = x\cos\theta + y\sin\theta$, $y' = -x\sin\theta + y\cos\theta$.

Basically, The catalyst kernels are beneficial in terms of increasing the SNR between the cover and the stego objects. They also act as regularizers to reduce the feature space dimensionality, thus helping facilitating the convergence of the network.

### 2.2 Feature Maps Computation

After obtaining the feature maps, each network layer will process these features in 3 layer in three steps: convolution, batch normalization and non-linear activation.

CNN's convolutional layers are characterized by an input map $I$, a bank of filters $K$ and biases $b$. In the case of images, we could have as input an image with a height parameter denoted by $H$, width denoted by $W$ and $C = 3$ channels (red, blue and green) such that $I \in \mathbb{R}^{H \times W \times C}$. Subsequently for a bank of $D$ filters we have $K \in \mathbb{R}^{k_1 \times k_2 \times C \times D}$ and biases $b \in \mathbb{R}^D$. The output from this convolution procedure is given as follows:

$$(I * K)_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \sum_{c=1}^{C} K_{m,n,c} \cdot I_{i+m,j+n,c} + b \tag{2}$$

In our case the input image is grayscale, as shown in Fig. (??) i.e single channel C = 1. The Eq. (2) will be transformed to:

$$(I * K)_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} K_{m,n} \cdot I_{i+m,j+n} + b \tag{3}$$

The Batch Normalization [11] step is applied right before each non-linear activation layer to normalize the feature map to zero-mean and unit-variance, and thus help the gradient descend back-propagation algorithm to avoid being trapped in local minima. It serves mainly the speed of the learning phase and improves the detection accuracy. *LeakyReLU* Fig. 3 (for Leaky Rectified Linear Units) is performed as an activation function at the end of each convolutional layer. The use of *LeakyReLU* helps us avoid expensive exponentials operations by *tanh* function for example and other problems like vanishing gradient problem by *sigmoid* or dying neurons by the simple *ReLU* function.

$$f(x) = \begin{cases} 0.1 \times x & \text{for } x < 0 \\ x & \text{for } x \geqslant 0 \end{cases}$$

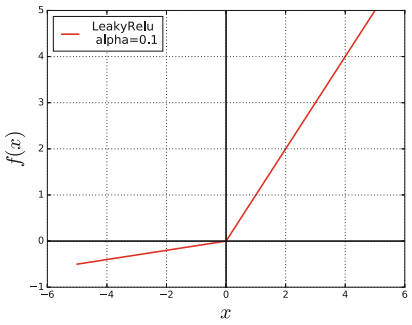



**Fig. 3.** *LeakyReLU* activation function (left) and its plot (right).

### 2.3 Classification and Optimization

The classification part of our network consists of a sequence of Three Dense layers, which are densely connected (also called fully connected) neural layers. The

first two layers include 200 neurons (per each). every neuron kernel is followed by **LeakyReLU** activation function (shown in Fig. 3).

The Third (and last) layer is a 2-way softmax layer, which means it will return an array of 2 probability scores (sum-ming to 1). Each score presents the probability of classification. In other words, the probability that a given input object belongs class1 (cover) or class2 (stego).

The fundamental trick in deep learning is to use the classification error between the real and the predicted class computed by a loss function (binary cross entropy in our case) in each iteration as a feedback signal to adjust the weight value gradually, in a direction presenting a lower loss rate.

Such adjustment is given through an optimizer, which is implemented fundamentally basing on what we call the back-propagation algorithm. We use RMSProp (for Root Mean Square Propagation) optimizer [12] with their default parameters.

### 2.4 Leaning from Scratch and Transfer Learning

For the highest payload (1 bpp in our case), we trained our network from scratch for 100 epoch to improve the detection accuracy and minimize the loss value as much as possible. The Fig. 4 shows the training model accuracy ans loss from scratch using WOW algorithm for stego embedding. This model cross 96% of detection accuracy after 100 epoch of training.

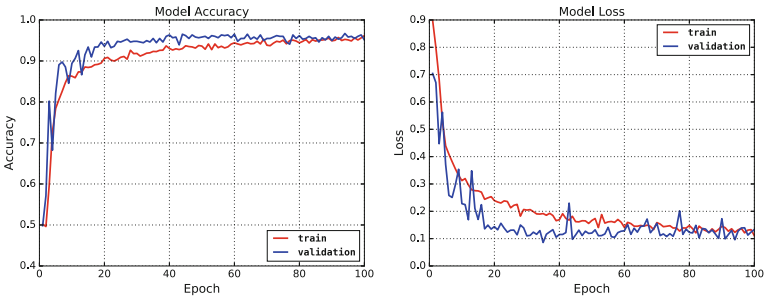
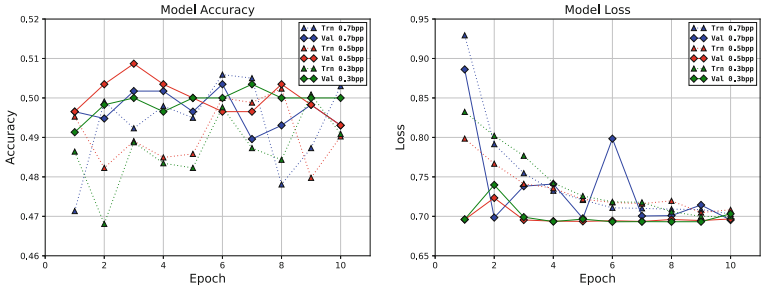


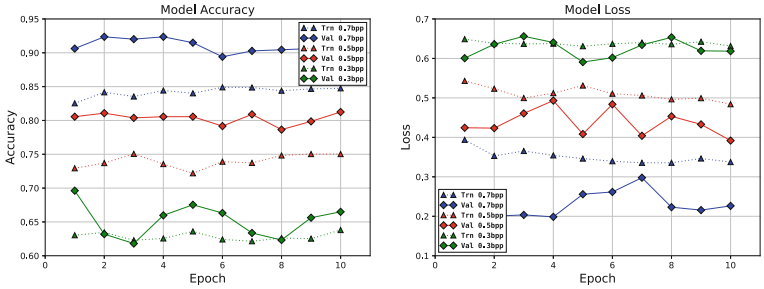

**Fig. 4.** Training model accuracy (right) and loss (left) during 100 epoch using WOW algorithm with high payload (1.0 bpp).

As shown in Fig. (5(a)), it is impossible to ensure a convergence with our model using learning from scratch with low payload such as 0.7 bpp, 0.5 bpp or 0.3 bpp. Thus, we choose to exploit the best weights of convolutional layers obtained by training on high payloads first, then we adjust these weights as an initialization to the training of our new dataset with lower payloads. This concept is called transfer learning and it is widely used in deep learning to minimize training time and ensure a rapid convergence of networks towards the best accuracy. Transfer learning gives better results in similar problems. As the

figure Fig. (5(b)) shows, the convergence of accuracy and loss becomes more stable and closer to logic. As the detection accuracy of training and validation remains stable, we limited the training process to 10 epoch.




(a) Training model accuracy (right) and loss (left) without transfer learning.




(b) Training model accuracy (right) and loss (left) with transfer learning.

**Fig. 5.** Difference between training with and without transfer learning using WOW algorithm with various payloads: 0.7 (blue curves), 0.5 (red curves) and 0.3 (green curves) bpp. (Color figure online)

## 3 Experimental Setup

### 3.1 Dataset

The used dataset is a part of the BOSSbase v1.01 [13] which contain 10,000 cover images of size $512 \times 512$. We just take the first 2400 images and resized to $128 \times 128$. Out of the 2400 pairs of images, 400 pairs (cover,stego) were split and left for testing to verify the performance of the proposed model; the rest of pairs were used in the training set. To train each CNN as a base learner, 1700 out of the 2000 training data are randomly shuffled at the beginning of each epoch, then the left 300 pairs were used for the validation. A mini-batch of 16 images (8 cover/stego pairs) was fed as in input for each iteration.

### 3.2 Software Platform

Training of the CNNs is performed on a Keras API using TensorFlow as backend. The implementation of the algorithms was performed on a computer CPU Intel Core $i3-2370M, 2.40\,\text{GHz} \times 4$, with Ubuntu 14.04 LTS.

## 4 Model Performance

For the test, We choose three highly undetectable stenographic algorithms in the spatial domain: WOW, S-UNIWARD, and HUGO. Similarly to the training phase, we prepared for the testing process different datasets with 1 bpp, 0.7 bpp, 0.5 bpp and 0.3 bpp payloads. In the following Table 1, we outline the classification accuracy measurement as well as the detection error obtained after the training of these different algorithms for several payloads.

**Table 1.** Model evaluation.

| | S-UNIWARD | | | | WOW | | | | HUGO | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Payload (bpp) | 1.0 | 0.7 | 0.5 | 0.3 | 1.0 | 0.7 | 0.5 | 0.3 | 1.0 | 0.7 | 0.5 | 0.3 |
| Accuracy | 0.96 | 0.93 | 0.88 | 0.73 | 0.96 | 0.92 | 0.83 | 0.67 | 0.96 | 0.91 | 0.84 | 0.69 |
| $P_E$ | 0.04 | 0.07 | 0.12 | 0.27 | 0.04 | 0.08 | 0.17 | 0.33 | 0.04 | 0.09 | 0.16 | 0.31 |

We notice that in Fig. 6 our method shows an efficient performance, in terms of classification, against S-UNIWARD algorithm for small payloads. The curve of WOW has the weakest detection almost for all payloads comparing to other approaches' curves.

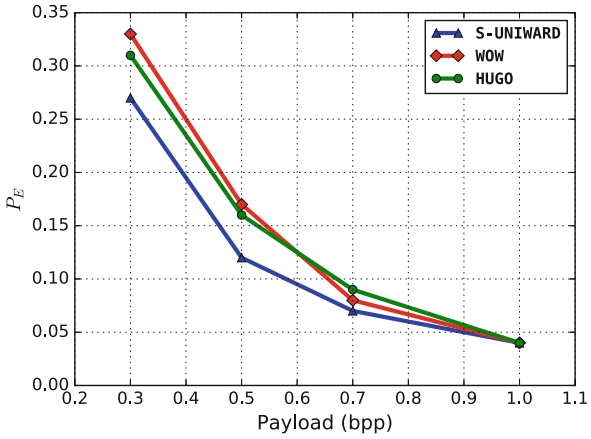



**Fig. 6.** Detection errors of the proposed classifier for the 3 state-of-the-art steganographic schemes (S-UNIWARD, WOW, HUGO) as a function of the embedded payload.

## 5 Conclusion

In this paper, on the one hand, we combined the robustness of the catalyst kernels to extract the residual noises in the aim of improving the difference between the cover and the stego images and on the other hand we exploited the technique of transfer learning which reduces the time of the convergence of our network specifically for small payloads. Provided experimental results show an interesting and a competitive performance, which can be the starting point of our future works intending mainly the establishment of accurate classification rates, not only for small sized images with small payloads, but for higher dimensions of the used input data.

## References

1. Pevný, T., Filler, T., Bas, P.: Using high-dimensional image models to perform highly undetectable steganography. In: Böhme, R., Fong, P.W.L., Safavi-Naini, R. (eds.) IH 2010. LNCS, vol. 6387, pp. 161–177. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16435-4_13
2. Holub, V., Fridrich, J.: Designing steganographic distortion using directional filters. In: 2012 IEEE International Workshop on Information Forensics and Security (WIFS), pp. 234–239. IEEE (2012)
3. Holub, V., Fridrich, J., Denemark, T.: Universal distortion function for steganography in an arbitrary domain. EURASIP J. Inf. Secur. **2014**(1), 1 (2014)
4. Qian, Y., Dong, J., Wang, W., Tan, T.: Deep learning for steganalysis via convolutional neural networks. In: Media Watermarking, Security, and Forensics, vol. 9409, p. 94090J (2015)
5. Couchot, J.-F., Couturier, R., Salomon, M.: Improving blind steganalysis in spatial domain using a criterion to choose the appropriate steganalyzer between CNN and SRM+EC. In: De Capitani di Vimercati, S., Martinelli, F. (eds.) SEC 2017. IAICT, vol. 502, pp. 327–340. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58469-0_22
6. Pibre, L., Pasquet, J., Ienco, D., Chaumont, M.: Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover sourcemismatch. Electron. Imaging **2016**(8), 1–11 (2016)
7. Xu, G., Wu, H.-Z., Shi, Y.Q.: Ensemble of CNNs for steganalysis: an empirical study. In: Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security, pp. 103–107. ACM (2016)
8. Fridrich, J., Kodovsky, J.: Rich models for steganalysis of digital images. IEEE Trans. Inf. Forensics Secur. **7**(3), 868–882 (2012)
9. Xu, G., Han-Zhou, W., Shi, Y.-Q.: Structural design of convolutional neural networks for steganalysis. IEEE Sig. Process. Lett. **23**(5), 708–712 (2016)
10. Chen, M., Sedighi, V., Boroumand, M., Fridrich, J.: JPEG-phase-aware convolutional neural network for steganalysis of JPEG images. In: Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security, pp. 75–84. ACM (2017)
11. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456 (2015)

12. Tieleman, T., Hinton, G.: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning. Technical report. https://zh.coursera.org/learn/neuralnetworks/lecture/YQHki/rmsprop-divide-the-gradient-by-a-running-average-of-its-recent-magnitude. Accessed 21 Apr 2017
13. Bas, P., Filler, T., Pevný, T.: Break our steganographic system: the ins and outs of organizing BOSS. In: Filler, T., Pevný, T., Craver, S., Ker, A. (eds.) IH 2011. LNCS, vol. 6958, pp. 59–70. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24178-9_5