Deep Learning-Based Steganalysis for Detection and Classification of Possible Hidden Content in Images

Mostafa A. Ahmad [1,*], Eftkhar Al-Qhtani[2], Ahmed H. Samak[1], Amr Ibrahim[1], Mourad Elloumi[1], Ali Ahmed [3]

[1]Department of Computer Science and Artificial Intelligence, College of Computing and Information Technology, University of Bisha, Bisha 61922, Saudi Arabia
[2] Department of Information Systems, College of Computing and Information Technology, University of Bisha, Bisha 61922, Saudi Arabia
[3]Information technology Department, Faculty of Computers and Information, Menoufia University, Egypt

Emails: mamoustafa@ub.edu.sa; aftkhar@ub.edu.sa; ahismail@ub.edu.sa; aemahmod@ub.edu.sa; melloumi@ub.edu.sa; ali.ahmed@ci.menofia.edu.eg

Abstract

Steganalysis can be defined as the science that addresses the process of identifying and detecting hidden information or data within various types of digital media. Recently, Deep Learning (DL) approaches have been employed to build steganalysis systems. However, the problem with steganalysis systems adopting a DL approach is their low accuracy and their need for effective datasets to be used for the training. In this paper, we introduce a DL-based Steganalysis system for the detection and classification of hidden content in images. Our system, called Steg-Analysis Convolutional Neural Network (SA-CNN), relies on a Convolutional Neural Network (CNN) and uses High Pass Filter (HPF) and extra-embedded data. We also propose a preprocessing-based data hiding method to increase the accuracy of SA-CNN in detecting hidden content. Therefore, this ensures the imperceptibility of images used for training SA-CNN. In addition, we use another CNN, called Malicious-Benign Classification CNN (MBC-CNN), that we have developed to classify the extracted hidden content into Malicious or Benign classes. Compared with existing systems, SA-CNN shows a better performance in terms of accuracy, under increased hiding rates ranging from 0.1 to 1.0 bpp, reaching 90%.

Keywords: Image Steganalysis; Deep Learning; Detection and Classification of Hidden Content

1. Introduction

Steganography can be defined as the science that addresses the process of hiding secret data within another data file. When compared with Cryptography, the basic objective of Steganography is to hide the communication itself [1,2,3]. Therefore, Steganography is performed at the communication level. In a steganography-based system, there are three main files, namely the cover-file (commonly, it is an image file), the secret data file, and the stego-file. Steganalysis can be defined as the science that addresses the process of identifying and detecting hidden information or data within various types of digital media. This purpose should be achieved even if there is a high level of similarities between the cover-file and the stego-file [4,5,6].

Steganography is considered one of the ways that threaten the security of networks. For instance, harmful content can be embedded inside transmitted images across the communication network. People use multimedia on a daily basis. With the diversity of multimedia, hacking operations vary, which makes someone wonder if this medium is protected or not for use. Consequently, the process of hiding data becomes a threatening factor for both the security of networks and the security of information itself. This is the motivation behind this research, which can be represented by the following question: Can we detect hidden data inside digital images? If so, how can we

DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024

recognize the type of hidden data? Researchers to help detect hidden data only present Steganalysis. The importance of Steganalysis comes from its valuable applications in real life.

The main goal of this study is to design and implement a Deep Learning (DL)-based steganalysis system with high accuracy in the prediction and classification of hidden content in images. To reach this goal, the following objectives are taken into account:

1. Establishing imperceptibility of images used to conduct the training phase.
2. Increasing the accuracy of the system to predict whether a given image is embedded with data or empty according to different hiding data rates.
3. Enabling the system to extract the hidden content, if there is any.
4. Qualifying the system to classify the extracted content into Benign or Malicious classes.

In this paper, we make the following main contribution:

- A first DL Convolutional Neural Network (CNN), called SA-CNN, to detect possible hidden content in an input image, through extracting features of the images.
- A second DL CNN, called MBC-CNN, classifies the extracted hidden content into Malicious or Benign classes. Let us note that in this paper, we mean by benign content a content that is a mere text while we mean by malicious content a harmful programming code.

The rest of this paper is organized as follows:

In Section 2, we make a literature review on Steganalysis approaches.

In Section 3, we present our DL-based image steganalysis approach and the corresponding CNNs, SA-CNN and MBC-CNN.

In Section 4, we make an experimental study on SA-CNN and MBC-CNN discuss the obtained results.

Finally, in section 5, we present the conclusion of this paper and provide some future work.

2. Related Work

Steganalysis approaches can be classified into two main categories: traditional and DL-based ones.

2.1. Traditional Steganalysis

There are different types of traditional Steganalysis:

(a) Signature-based Steganalysis

(b) Statistical Steganalysis

(c) Spread spectrum Steganalysis

(d) Transform domain Steganalysis

The key idea of signature-based Steganalysis is to search for signature patterns to check the presence of an embedded message, where specific bytes of the image are used. In the work described in [10], Discrete Cosine Transform (DCT) coefficients are used to extract the quantization matrix of all image blocks then it was compared with the standard JPEG quantization table to check if there are any incompatible blocks in the image.

The objective of statistical Steganalysis [11,12] is to analyze the embedding procedure and determine statistics that can be modified as a result of the embedding process. The authors in [13] propose a technique, based on Fourier Transform of image histogram, to exploit the properties of the center of mass of the Histogram Characteristic Function (HCF), which is the first order moment. Many of the transform domains methods were employed, like DCT, Discrete Wavelet Transform (DWT) and Fast Fourier Transform (FFT), for embedding information in transform coefficients of the cover-images [14].

Recently, DL-based approaches are effective in Steganalysis. Indeed:

Kim et al. [15] propose a Steganalysis system, based on a CNN, to add additional data to the original stego-images for highlighting possible hidden content. They used BOSSBase dataset to train their system, and achieved an accuracy rate of 80.05%.

DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024

The authors of the work [16] propose a CNN, called SFRNet, made-up of two blocks: The first one, called RepVgg block, is used to accelerate the inference, and the second one, called SE block, is used to improve the detection accuracy. The achieved accuracy rates of SFRNet are 72.5% and 89.6% at 0.2 bits per pixel (bpp) and 0.4 bpp respectively.

Yuwei et al. [17] proposes a CNN to extract subtle features that indicate the existence of hidden content. The achieved accuracy rates are 76.2% and 88.7% at 0.2 bpp and 0.4 bpp, respectively, when BOSSBase1.01 is used. The accuracy rates achieved are 80.6% and 89.2 % at 0.2 bpp and 0.4 bpp, respectively, when the BOWS2 dataset is used.

In the study presented in [18], the authors present a CNN, called GBRAS-net. That includes a preprocessing phase using filter banks to enhance steganographic noise, a feature extraction phase using depth wise and separable convolutional layers and skip connections. BOWS 2 and BOSSbase 1.01 datasets were used to evaluate the performance of GBRAS-net. The achieved accuracies are good. For example, the achieved accuracy rates are 74.6% and 84.5% at 0.2 bpp and 0.4 bpp, respectively, when BOSSbase 1.01 dataset is used.

Lwowski et al. [19] develop Neural Spatial Rich Models (NSRM) that use Spatial Rich Model (SRM) [46] for feature extraction, Principal Component Analysis (PCA) for dimensionality reduction, and Deep NNs (DNNs) to perform JPG image steganalysis against four different well-known steganography algorithms at five different embedded rates.

In [20], FractalNet is a DNN used for steganalytic detection. FractalNet's architecture allows transition during training from shallow Neural Networks (NN) to DNN, thereby achieving better detection performance.

Pant et al. [21] adopt Transfer Learning in a pre-trained VGG16 model to classify malware presented as grayscale images. They achieved an accurate rate of 88.40%.

The authors in [22] propose CNN for malware classification based on custom CNN, AlexNet, VGG-16, ResNet-50, and Inceptionv3. They tested their CNN using the Malimg dataset and achieved an accurate rate of 98.90%.

The authors of the work [23] examined the J-UNIWARD approach to build a steganalysis system to detect hidden data embedded. They trained their model using BOSSBase and ImageNet datasets and they obtained a detection error rate of 35.98% and 16.8% respectively.

The work proposed in [24] introduces an Automatic Steganography Distortion Learning (ASDL) framework with Generative Adversarial Network (GAN) that can automatically learn embedded change probabilities for every pixel in a given spatial cover image. The model was trained using the BOSSBase dataset, and a detection error rate of 33.02% was obtained.

Based on the idea of amplifying the noise that is caused by steganography, the authors of the work [25] proposed a CNN-based steganalysis system. They used high pass filtering to initiate the trainable filters instead of using a random way and also to distinguish noise. They trained their model using BPPS dataset and they achieved a detection error rate of 27.30%.

The authors of the work [26] propose a CNN called Yedroudj-Net. They rely on a pre-processing filter bank and a truncation activation function, five convolutional layers with a batch normalization associated with a scale layer. They used an augmented dataset of the BOSSBase dataset to enhance the training of the CNN model and they achieved a detection error rate of 27.80%.

Tsang et al. [27] propose YeNet network to address the problem of different resolutions. The key idea of their work is to train the steganalysis system on small-resolution images and be able to detect hidden contents on high-resolution images. They achieved a detection error rate of 14.45% when using the BOSSBase dataset.

The objective of the work done in [28] is to strengthen the noise caused by the hiding process of steganography so that stego-images can be recognized from the cover objects. Their CNN is trained using the BOSSBase dataset and a detection error rate of 23.33% was obtained.

Deep Residual learning-based Network (DRN) is proposed in [29], it has two distinguished features when compared with other CNN-based approaches. The first feature is the number of network layers that benefit from capturing the complex statistical information held by images, and the second feature is keeping the signal coming from stego-images, which in turn benefits enhancing the accuracy of determining cover objects from stego ones. DRN was trained on the BOSSBase database, and the results show that the accuracy in terms of error rate is 10.4%.

The problem addressed by the authors of [30] is related to weak distortion caused by the hiding process of data, which in turn leads to low detection. Their CNN mainly depends on strengthening the hidden message by adopting

DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024

pre-processing filters. BOSSBase dataset was used to train their CNN and a classification rate of 79.56% was obtained.

Most of the recent proposed models in Steganalysis are summarized in [31, 32, 33, 34]. Although much work has been done using DL in the Steganalysis research field, still the lack of effective existing datasets used for training is a problem faced by many researchers and hence the problem of low accuracy achievement.

In what follows, we present our DL-based image steganalysis approach and the corresponding CNNs, SA-CNN, and MBC-CNN.

3. Our DL-Based Image Steganalysis Approach

The goal of our DL-based steganalysis approach is to protect network users from malicious codes that can intrude into their machines, taking into consideration that malicious codes can be hidden within innocent images. To reach this goal, our DL-based approach operates in two stages:

(i) During the first stage, a training process is performed on a specific dataset of images to be able to detect possible hidden content in an input image, through extracting features of the images So, a DL CNN, called Steg-Analysis Convolutional Neural Network (SA-CNN), is employed to achieve this task.

(ii) During the second stage, a training process is performed on the extracted features of images, obtained thanks to SA-CNN, to be able to classify the possible extracted hidden content into malicious or benign content. This task is achieved thanks to a second DL CNN, called Malicious-Benign Classification CNN (MBC-CNN).

Once these two stages are achieved, it is possible to protect network users from malicious codes hidden within harmful innocent images.

Therefore, our DL-based steganalysis approach classifies input images into two main classes, where the second one is partitioned into two subclasses:

(a) Class C_1: It is the class of clean images, i.e., images with no embedded data.

(b) Class C_2: It is the class of embedded images, i.e., images with embedded data.

This class is further partitioned into two subclasses:

(b1) Subclass C_{2a} is the subclass of images embedded with malicious content.

(b2) Subclass C_{2b} is the subclass of images embedded with benign content.

Figure 1 illustrates the general framework of our DL-based approach.

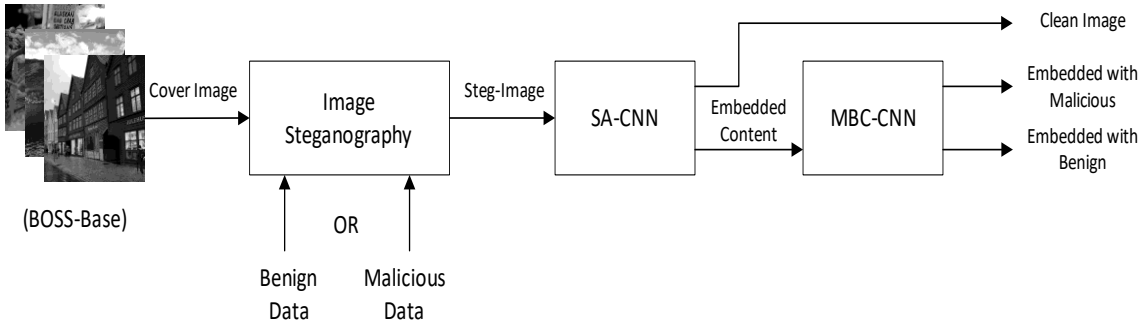


Figure 1. General framework of our approach.

3.1 SA-CNN Architecture

The general architecture of SA-CNN consists of a series of convolutional layers followed by pooling ones. CNNs differ from each other according to the number of layers that are used. Figure 2 illustrates the architecture of SA-CNN.

DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024

	Name	Type	Activations	Learnable Properties
1	Imageinput 512×512×1 images	Image input	512(s)×512(s)×1(c)×1(B)	-
2	Conv_1 1 5×5 convolutions	2-D Convolution	508(s)×508(s)×1(c)×1(B)	Weights 5×5×1×1 Bias 1×1×1
3	Batchnorm_1 Batch normalization	Batch Normalization	508(s)×508(s)×1(c)×1(B)	Offset 1×1×1 Scale 1×1×1
4	Layer_1 Hyperbolic tangent	Tanh	508(s)×508(s)×1(c)×1(B)	-
5	Conv_2 16 5×5 convolutions	2-D Convolution	504(s)×504(s)×16(c)×1(B)	Weights 5×5×1×16 Bias 1×1×16
6	Batchnorm_2 Batch normalization	Batch Normalization	504(s)×504(s)×16(c)×1(B)	Offset 1×1×16 Scale 1×1×16
7	Layer_2 Hyperbolic tangent	Tanh	504(s)×504(s)×16(c)×1(B)	-
8	Conv_3 16 5×5 convolutions	2-D Convolution	500(s)×500(s)×16(c)×1(B)	Weights 5×5×16×16 Bias 1×1×16
9	Batchnorm_3 Batch normalization	Batch Normalization	500(s)×500(s)×16(c)×1(B)	Offset 1×1×16 Scale 1×1×16
10	Layer_3 Hyperbolic tangent	Tanh	500(s)×500(s)×16(c)×1(B)	-
11	Conv_4 32 5×5 convolutions	2-D Convolution	496(s)×496(s)×32(c)×1(B)	Weights 5×5×16×32 Bias 1×1×32
12	Batchnorm_4 Batch normalization	Batch Normalization	496(s)×496(s)×32(c)×1(B)	Offset 1×1×32 Scale 1×1×32
13	Relu_1 ReLU	ReLU	496(s)×496(s)×32(c)×1(B)	-
14	Conv_5 32 5×5 convolutions	2-D Convolution	492(s)×492(s)×32(c)×1(B)	Weights 5×5×32×32 Bias 1×1×32
15	Batchnorm_5 Batch normalization	Batch Normalization	492(s)×492(s)×32(c)×1(B)	Offset 1×1×32 Scale 1×1×32
16	Relu_2 ReLU	ReLU	492(s)×492(s)×32(c)×1(B)	-
17	Fc_1 32 fully connected	fully connected	1(s)×1(s)×32(c)×1(B)	Weights 32×7746048 Bias 32×1
18	Fc_2 16 fully connected	fully connected	1(s)×1(s)×16(c)×1(B)	Weights 16×32 Bias 16×1
19	Fc_3 2 fully connected	fully connected	1(s)×1(s)×2(c)×1(B)	Weights 2×16 Bias 2×1
20	Softmax Softmax	Softmax	1(s)×1(s)×2(c)×1(B)	-
21	Classoutput Crossentropyex	Classification output	1(s)×1(s)×2(c)×1(B)	-

Figure 2. Architecture of SA-CNN.

SA-CNN consists of five convolutional layers. In addition, there are five batch normalization ones, after each convolutional layer, used to speed up the process of training of SA-CNN. In addition, there are three fully connected layers.

Therefore, if we consider that the size of the input image is 512×512 grayscale of 8-bit depth, we have the following layers:

- The first convolutional layer uses only one filter of size 5×5.
- The second and third convolutional layers use 16 filters.
- In addition, the fourth and fifth convolutional layers use 32 filters.

All the filters are of the same size as the first one. This is to set similar areas of interest for feature extraction each time.

The following activation functions are used:

- The Tanh activation function is used after the first and the second convolutional layers. It takes any real value as input and output values in the range -1 to 1. This function is mainly used for classification between two classes.
- The ReLU activation function is used after the third, fourth, and fifth convolutional layers. This function is used to solve the vanishing gradients issue. ReLU is a nonlinear function, $f(x) = max(0, x)$, which sets all negative values to zero.
- Finally, the softmax activation function is used for classification purposes.

The following algorithm represents the pseudo-code of SA-CNN.

DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024

Algorithm 1: Steg-Analysis CNN (SA-CNN)

Input: Image_HPF

Output: Image_HPF.Classification_Label

Steps:

Training_Image = Get_Image_From_Training_DataSet()

[Width, Hight]=Get_Size(Training_Image)

Layers=Set_Layers()

Testing_Image=Get_Image_From_ Testing _DataSet()

Option=Set_Options()

Training_Network=Train_Network(Training_Image,Layers,Options)

Classification_Labels=[Embedded, Clean]

Image_HPF.Classification_Label=Classify(Training_Network, Testing_Image, Classification_Labels)

YValidation=Testing_Image.Labels

Accuracy=sum(Image_HPF.Classification_Label==Testing_Image.Classification_Label)/numel(Testing _Image. Classification_Label)

3.2 MBC-CNN Architecture

MBC-CNN consists of six convolutional layers as well as six batch normalization layers. There are also six pooling average layers followed by three fully connected layers.

The following activation functions are used:

- The activation function ReLU is used after each batch normalization layer.
- Moreover, the softmax activation function is used for classification purposes.

The following algorithm represents the pseudo-code of MBC-CNN. Let us recall that in this algorithm, we mean by benign content that is a mere text while we mean by Malicious a harmful programming code.

Algorithm 2: Malicious-Benign Classification CNN (MBC-CNN)

Input: Image_HPF

Output: Image_HPF.Classification_Label

Steps:

Training_Image=Get_Image_From_Training_DataSet()

[Width, Hight]=Get_Size(Training_Image)

Layers=Set_Layers()

Testing_Image=Get_Image_From_ Testing _DataSet()

Option=Set_Options()

Training_Network=Train_Network(Training_Image,Layers,Options)

Classification_Labels=[Malicious, Benign]

Image_HPF.Classification_Label=Classify(Training_Network, Testing_Image, Classification_Labels)

YValidation=Testing_Image.Labels

Accuracy=Sum(Image_HPF.Classification_Label==Testing_Image.Classification_Label)/numel(Testing _Image. Classification_Label).

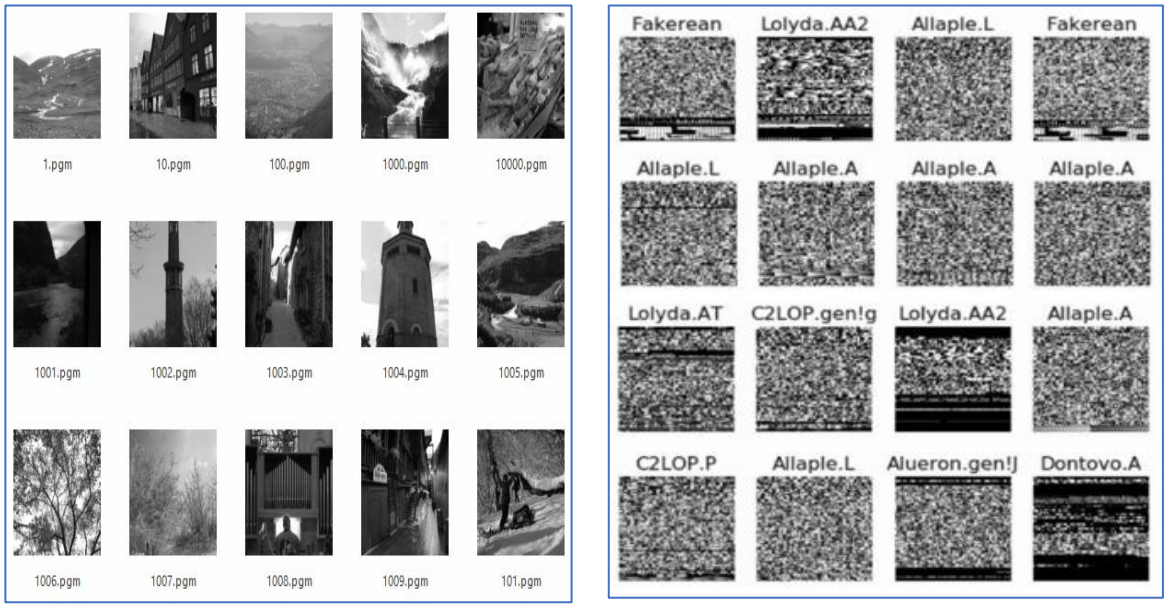
DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024

3.3 Datasets

The main purpose of this step is to prepare a suitable environment by preparing the datasets and training the SA-CNN. In this study, the two datasets used are BOSSBase [47] and Malimg [48] as cover and malicious images datasets, respectively. Figure 3 shows samples from the BOSSBase and Malimg datasets

BOSSBase dataset contains 10,000 grayscale images of a size of 512×512. The texts used to be embedded into cover-images are selected from the project Gutenberg [9]. So, an image in the BOSSBase dataset is either embedded, using Least Significant Bits (LSBs), with a benign text or not embedded with any text, i.e., it's a clean image.

Malimg dataset contains 9339 grayscale images categorized into 25 classes. It is worth mentioning that the Malimg dataset is highly unbalanced. Indeed, Class 2 and Class 3 dominate the dataset [35, 36, 37], where more than 30% of images belong to Class 2 (Allaple.A) and 17% belong to Class 3 (Allaple.L). This is the main reason behind selecting this dataset as it is considered a typical environment to manipulate the imbalanced data issue. From a visualization perspective, embedded images can be represented by four sections, which are .text, .rdata, .data, and .rsrc [38].



(a) Sample images from BOSSBase dataset. (b) Sample images from Malimg dataset.

Figure 3. Sample images from BOSSBase and Malimg datasets.

3.4 Data Preprocessing

The purpose of the data preprocessing stage is to hide benign (mere text) or malicious content (harmful programming code) within cover-images to build a preprocessing dataset, called Preprocessed Data Set (PDS). The PDS dataset will be used for training and testing.

Actually, there are four main steps applied to build a stego-images dataset:

(i) During the first step, the Least Significant Bit (LSB) hiding technique [39, 40] is applied.

(ii) During the second step, images are embedded with hidden contents, whether malicious or benign ones, to obtain stego-images.

(iii) The generated stego-images are then gathered together to form a stego-images dataset.

(iv) Finally, to apply the cross-validation principle, the stego-images dataset is split into two parts, the training and testing parts. In the case of the PSD dataset:

(a) The first part, which represents 70% of PSD, i.e., 105 stego-images, is used to train SA-CNN and MBC-CNN.

(b) The second part, which represents 30% of PSD, i.e., 45 stego-images, is used to test SA-CNN and MBC-CNN.

DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024

3.5 Establishing Imperceptibility

To measure the robustness of the PDS dataset, two aspects are considered:

(a) The first aspect concerns the benign content. That is why the Peak Signal to Noise Ratio (PSNR) values are measured.

Indeed, PSNR measures the amount of distortion caused after hiding a content [41]. The higher PSNR value the higher imperceptibility of hidden content. PSNR is given by the following equation:

$$\mathrm{PSNR} = 10 \times \log_{10}[\frac{\mathrm{CoverI}^{\,2}}{\mathrm{MSE}}] \tag{1}$$

Where: CoverI is the cover image, and MSE is the Mean Square Error. MSE is given by:

$$\mathrm{MSE} = \frac{1}{[\mathrm{row} \times \mathrm{colomn}]^2} \times \sum_{\mathrm{i}=0}^{\mathrm{k}} \sum_{\mathrm{j}=0}^{\mathrm{d}} (\mathrm{PintensiyCI_{ij}} - \mathrm{PintensiySI_{ij}})^2 \tag{2}$$

(b) The second aspect concerns the malicious content. That is why the Structural Similarity (SSIM) values are measured. Indeed, SSIM measures the variation of structural information between the cover-image and the stego-image [42]. SSIM is given by the following equation:

$$\mathrm{SSIM}\left(\mathrm{img}_{\mathrm{AADS}}^{\mathrm{cover}}.\,\mathrm{img}_{\mathrm{AADS}}^{\mathrm{stego}}\right) = \left[\mathrm{lum}\left(\mathrm{img}_{\mathrm{AADS}}^{\mathrm{cover}}.\,\mathrm{img}_{\mathrm{AADS}}^{\mathrm{stego}}\right)\right]^{\grave{e}} \times \left[\mathrm{con}\left(\mathrm{img}_{\mathrm{AADS}}^{\mathrm{cover}}.\,\mathrm{img}_{\mathrm{AADS}}^{\mathrm{stego}}\right)\right]^{\tilde{v}} \times \left[\mathrm{str}\left(\mathrm{img}_{\mathrm{AADS}}^{\mathrm{cover}}.\,\mathrm{img}_{\mathrm{AADS}}^{\mathrm{stego}}\right)\right]^{\hat{z}} \tag{3}$$

where ($\grave{e}, \tilde{v}, \hat{z} > 0$) are parameters used to control the luminance, contrast, and structural elements, respectively. The values of SSIM fall within the range [0,1], where 0 indicates that the quality of the stego-image is too bad while 1 indicates that the quality of the stego-image is perfect.

Let's note that the statistical analysis, based on the average values of PSNR, indicates that the boundaries of PSNR are considered normal if they are between 51 and 61, either in the case of malicious content or a benign one. This means that there is almost perfect match between the stego-images and the corresponding cover-images [43]. In addition, when analyzing the values of PSNR of each single level of hiding rate, it is observed that the value of PSNR is higher in the case of hiding malicious content compared with the case of hiding benign one. This means that the nature of malicious content (harmful programming code) differs from the nature of the benign content (mere text). Actually, the values of pixels in images are correlated, which are similar to each other or approximately close, while in the text, the correlation of consequent letters is rare.

3.6 Training Our CNNs

During this stage, SA-CNN and MBC-CNN are trained. The training process requires feature extraction to be the base of training. Features extraction, in both CNNs, goes through a series of convolutional layers followed by pooling ones.

A filter/kernel scans the input image in a convolutional fashion. After the first convolution, represented by the first convolutional layer, features are extracted. Then, the extracted features are pooled through a pooling layer. In each CNN, there are five convolutional layers followed by four pooling ones. After extracting the final features of the input image, both CNNs are trained on these features to have the ability to classify images. For actual classification from a mathematical point of view, the softmax function is used.

3.7 Testing Our CNNs

During this phase, SA-CNN and MBC-CNN are tested on the testing part of the PDS dataset. Each image, in the testing part of the PDS dataset, is first entered in SA-CNN. If this image is embedded with a content, MBC-CNN classifies the embedded content in malicious or benign classes.

4. Results and Discussions

In this section, we make an experimental study on SA-CNN and MBC-CNN discussing the results obtained.

4.1 Experiments Setup

Both SA-CNN and MBC-CNN are implemented in MATLAB 2020b, with the required toolboxes, on a laptop equipped with hexa-core Intel(R) Core (TM) i7-9750HF CPU whose base frequency is 2.60 GHz. The RAM capacity is 16 GB and the hard drive capacity is 1TB. The operating system is Microsoft Windows 10 Home (x64). The parameters that adjust the value of the hiding ratio are set by default to 0.1, with the ability to change them

DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024

for conducting different experiments. In addition, the size of the used filters in SA-CNN and MBC-CNN is set by default to [3×3], with the ability of changing it for further experiments.

4.2 Evaluation Metrics

The confusion matrix helps to visualize a classification model performance by comparing predicted values against actual values for a dataset. Table 1 represents our confusion matrix based on the following accuracy metrics:

(a) True Positives (TP): Number of positive images correctly recognized by SA-CNN as images with embedded content.

(b) True Negatives (TN): Number of negative images correctly recognized by SA-CNN as images with no embedded content.

(c) False Positives (FP): Number of images incorrectly recognized by SA-CNN as images with embedded content.

(d) False Negatives (FN): Number of negative images incorrectly recognized by SA-CNN as images with no embedded content.

Table 1: Confusion matrix of evaluation of the model.

Predicted class		**Actual class**	
		Positive	**Negative**
	Positive	TP	FP
	Negative	FN	TN

The performance metrics such as accuracy, sensitivity, and error rate metrics can be derived from the confusion matrix. For a given classifier, the accuracy can be calculated by:

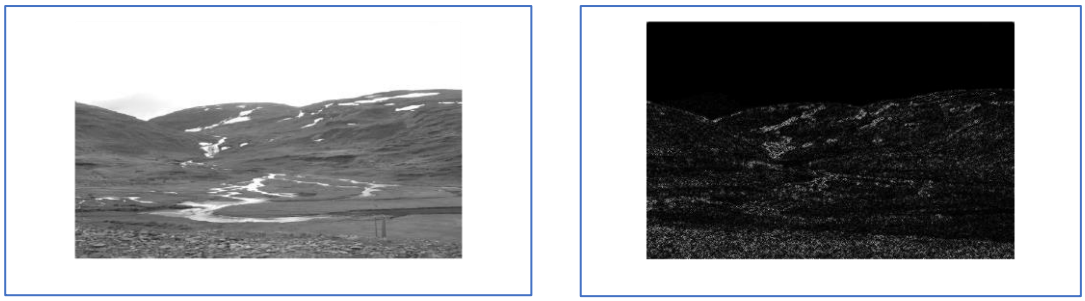
$$\text{accuracy} = \frac{(\text{TP}+\text{TN})}{\text{number of all images}} \tag{4}$$

4.3 Data Hiding Result

Figure 4 represents a sample of the hiding phase after applying High Pass Filter (HPF) [49]. As shown in this figure, when low amount of data (for example, 0.1 bpp) is hidden in the original image, it becomes impossible for the human eye to see the difference between the original image and the stego-image. However, if we use a high amount of data (for example, 5.0 bpp) hidden in the original image, we notice that the stego-image is distorted in comparison to the original image.

Figure 4 also clearly shows that applying HPF enables us to differentiate between the images that have many details, or features, and the images that are close to static, i.e., with a few features. This is actually a reflection of the task of the HPF because it controls the contrast of the images. In some details, the original image that represents a building generated a high-featured (high frequency) image after applying HPF. That is because it originally has details of high frequency. In contrast, the image that represents a natural view (Small Mountain) generates an image that has very few features after applying the HPF. This is because that, originally, this image has a low frequency of intensity waves.

DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024



(a) Stego-images before applying HPF (b) Stego-images after applying HPF

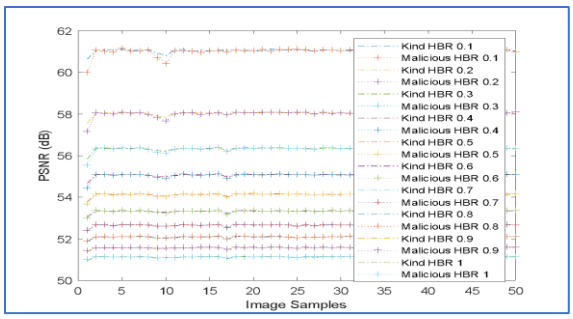
Figure 4. Image samples before and after applying HPF at hiding rate of 0.5 bpp.

4.4 Imperceptibility Evaluation

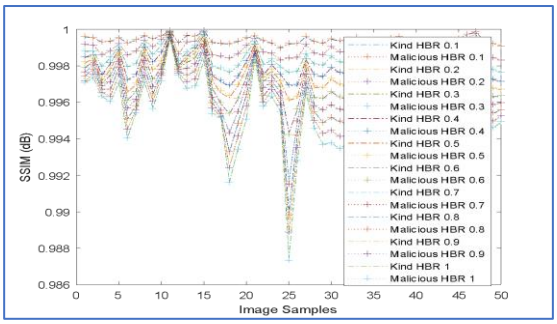
The statistical analysis, based on the average values of the SSIM, shows that the boundaries of the SSIM are approximately perfect, which are between 0.995 and 0.998, either in the case of hiding malicious or benign content. This means that there is almost perfect matching between the stego-images and their corresponding cover-images from a structural point of view.

When it comes to analyzing the values of SSIM of each single level of hiding rate, it is observed that the value of SSIM is higher in the case of hiding a malicious content compared with the case of hiding a benign one. The reason behind this is that the nature of malicious content (harmful programming code) differs from the nature of the content (mere text). Actually, the values of pixels in images are approximately close while in text there are different series of letters. This in turn leads to a little bit of effect on the structure of the image when hiding malicious content.

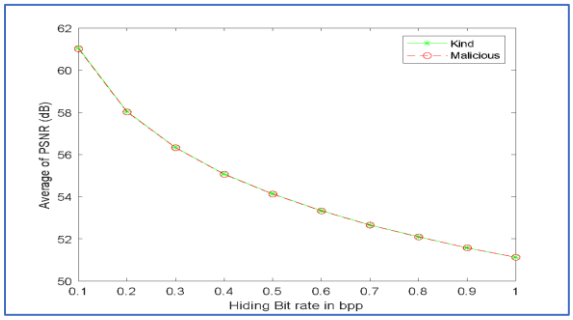
Figure 5(a) and Figure 5(b) show the values of the PSNR and SSIM for hiding benign and malicious contents for 50 random images at hiding ratio increases from 0.1 to 1.0 bpp with a step of 0.1. Figure 5(c) and Figure 5(d) show the average values of the PSNR and SSIM for both hiding malicious and benign content respectively.



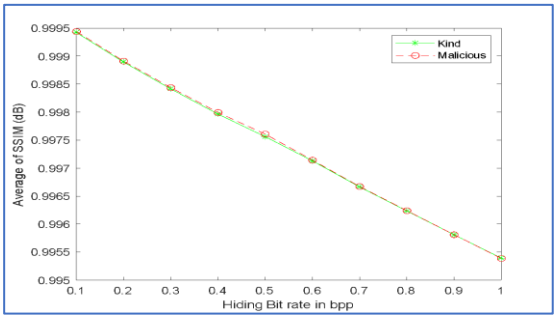
(a) PSNR values



(b) SSIM values



(c) PSNR average



(d) SSIM average

Figure 5. PSNR and SSIM values and averages for hiding benign and malicious content at different hiding rates.

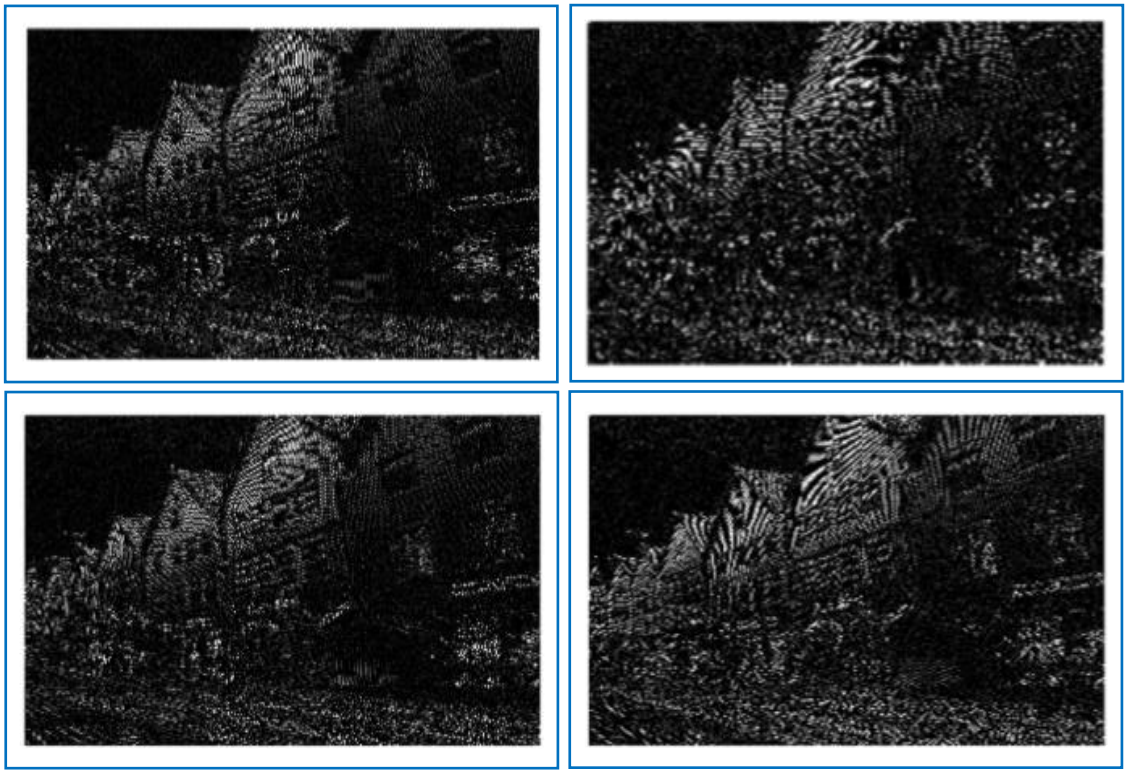
DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024

4.5 SA-CNN Performance Evaluation

To evaluate the performance of SA-CNN, different hiding rates were adopted. Table 2 highlights the accuracy of SA-CNN at hiding rates ranging from 0.1 to 1.0 bpp. We can notice from this table that the accuracy of SA-CNN increases as the hiding rate increases. This is due to the increase in content for, then the detecting accuracy increases. Consequently, the spot that represents a signal about existing hidden content is highlighted with an increased hiding rate. The average of the achieved accuracy gives a fair statistical measure of the high accuracy of SA-CNN, average = 77.10. Figure 6 shows samples of SA-CNN results at a hiding rate of 0.5 bpp.

Table 2: SA-CNN model accuracies at different hiding rates.

Hiding Rate (bpp)	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
Accuracy (%)	60	66.33	70	73.33	76.67	79.7	82.65	84.33	88	**90**



(a) Stego-images after applying HPF (b) SA-CNN Results

Figure 6. SA-CNN's output at hiding rate of 0.5 bpp.

To evaluate the performance of SA-CNN, we have compared the accuracy of SA-CNN with the one of the most recently published CNN, called Simulated Dual-CNN [45].

Table 3 represents a comparison between the accuracy of SA-CNN and the-one of Simulated Dual-CNN, at hiding rates of 0.1, 0.5 and 1.0 bpp. As shown in this table, SA-CNN outperforms Simulated Dual-CNN. Indeed, SA-CNN achieves accuracies of 60%, 76.67%, and 90% at hiding rates of 0.1 bpp, 0.5 bpp, and 1.0 bpp respectively. Whereas, Simulated Dual-CNN, for the same hiding rates, achieves accuracies of 50%, 66.67%, and 80% respectively. The reason behind the high accuracy of SA-CNN is related to using HPF that supports suitable size filters in convolutional layers and other layers that contribute a lot to highlight a possible hidden content to be detected.

Figure 7 illustrates the superiority of SA-CNN, in terms of accuracy, when compared with-Simulated Dual-CNN.

Figure 8 shows the training accuracy and loss of SA-CNN and Simulated Dual-CNN at the hiding rate of 0.5 bpp.

DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024

Table 3: Accuracy of SA-CNN compared with the-one of Simulated Dual-CNN [45].

Hiding Rate (bpp)	CNN Network	Accuracy (%)
0.1	Simulated Dual-CNN [45]	50
	SA-CNN	**60**
0.5	Simulated Dual-CNN [45]	66.67
	SA-CNN	**76.67**
1.0	Simulated Dual-CNN [45]	80
	SA-CNN	**90**

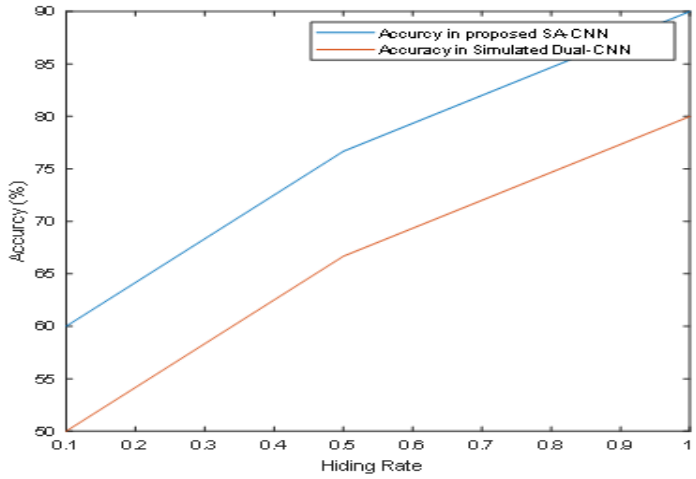
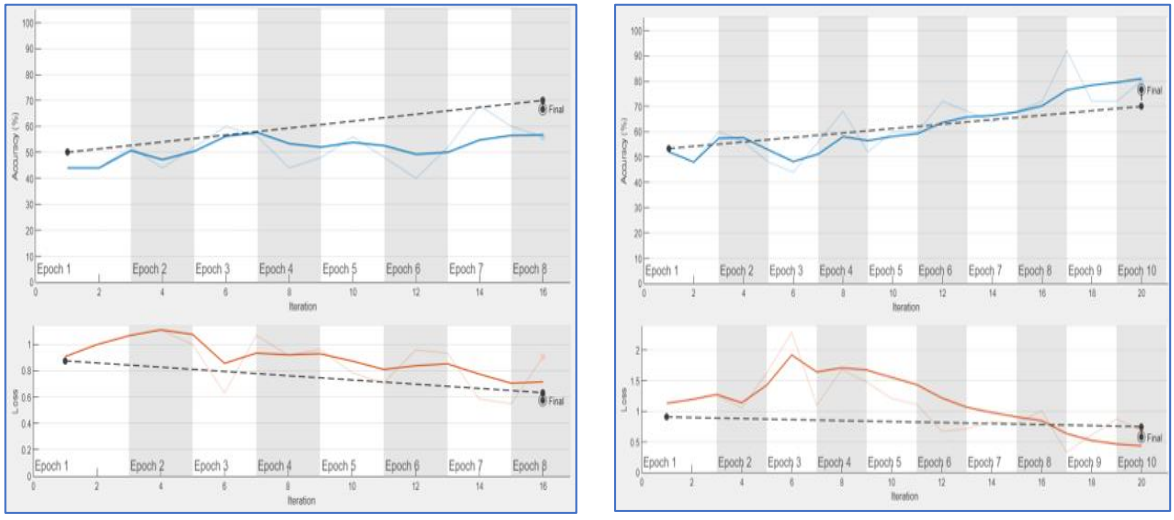


Figure 7. SA-CNN accuracy compared with the one of simulated dual-CNN model [45], at hiding rates ranging from 0.1 to 1.0 bpp.



(a) Simulated Dual-CNN [45] (b) SA-CNN

Figure 8. Training accuracy and loss of SA-CNN, at hiding rate of 0.5 bpp, compared with the-one of simulated dual-CNN [45].

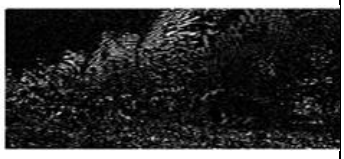
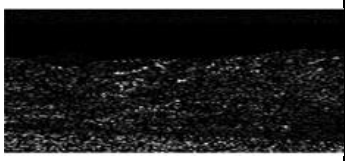
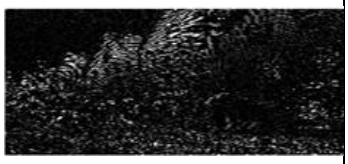
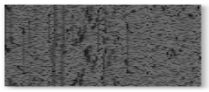
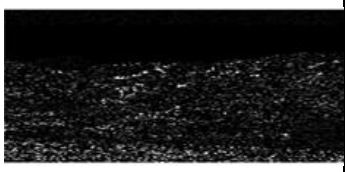
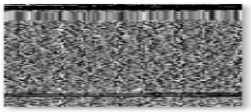
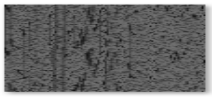
DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024

4.6 Hidden Data Extraction

After extracting the hidden content from the stego-image, thanks to SA-CNN, this hidden content is then saved in one of two files: Malicious images file or Benign images one, to prepare them for the training of MBC-CNN.

Table 4 shows examples of extracted hidden contents at hiding rates of 0.1 and 0.5 bpp.

Table 4: Hidden data extraction results at different hiding rates.

Hiding Rate (bpp)	Embedded stego-image after applying HPF	Extracted hidden contents	
		Malicious content	Benign content
0.1		 Img_Malicious_hbpp_0.1_1	 Img_Kind_hbpp_0.1_1
		 Img_Malicious_hbpp_0.1_2	 Img_Kind_hbpp_0.1_2
0.5		 Img_Malicious_hbpp_0.5_1	 Img_Kind_hbpp_0.5_1
		 Img_Malicious_hbpp_0.5_2	 Img_Kind_hbpp_0.5_2

4.7 MBC-CNN Performance

The extracted hidden contents, obtained thanks to SA-CNN, will be used to train and test MBC-CNN.

Figure 9 illustrates accuracy rates of MBC-CNN when classifying malicious and benign contents for 10 epochs.

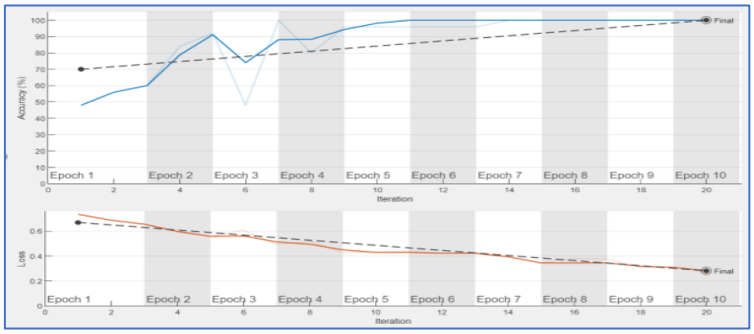
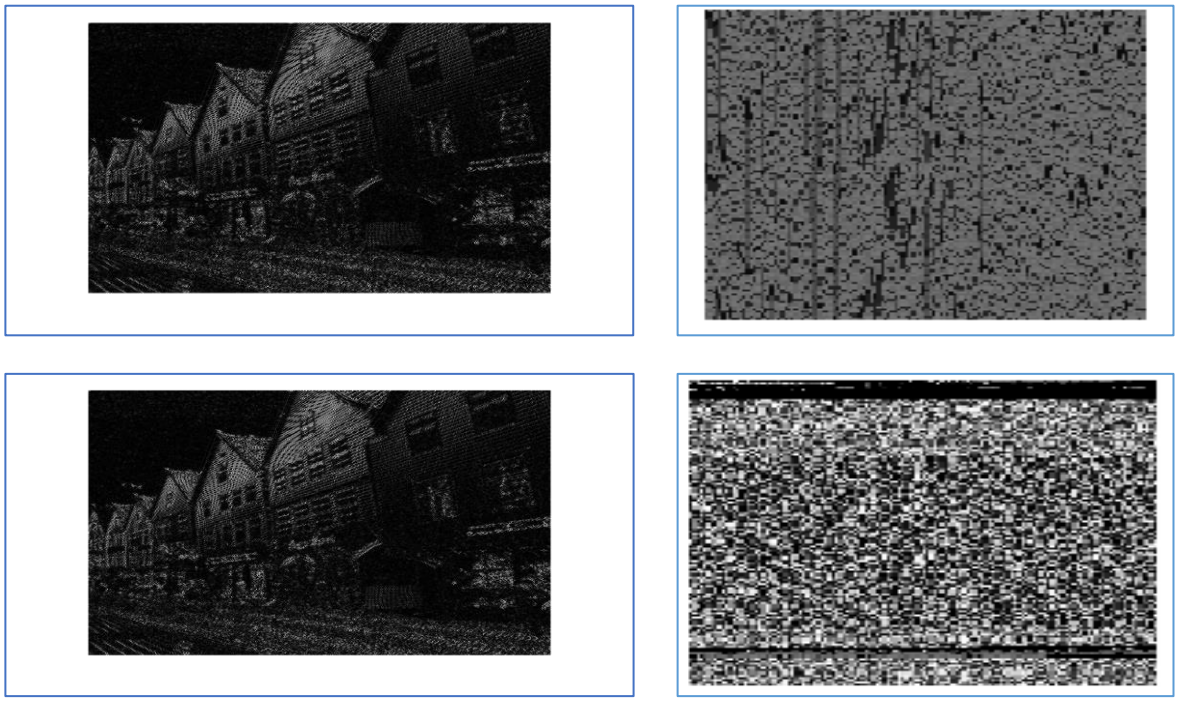


Figure 9. Accuracy of MBC-CNN at a hiding rate of 0.5 bpp.

As shown in Figure 9, the accuracy of MBC-CNN can reach 100%. The justification of this high training accuracy is that MBC-CNN is trained on different features that represent malicious or benign content. In other words, the features extracted, thanks to SA-CNN, of malicious content differ from those extracted from benign ones. This consequently means that the training process, in the malicious case or in the benign one, differs. Which in turn

DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024

leads to high contrast between malicious content and benign one. Therefore, MBC-CNN has achieved high accuracy in the process of content classification. Actually, this justification is supported visually in Figure 10. Indeed, as shown in Figure 10, a clear difference can be noticed even by the human eye between malicious contents and benign ones.



(a) Stego-Images

(b) MBC-CNN Outputs. Benign Data (Up), Malicious Data (Down)

Figure 10. MBC-CNN results at a hiding rate of 0.5 bpp.

5. Conclusion

In this paper, we introduced a Deep Learning (DL)-based steganalysis system with high accuracy in the prediction and classification of hidden content in images. Our system is mainly made up of two CNNs:

(a) The first CNN, called Steg-Analysis Convolutional Neural Network (SA-CNN), is a Convolutional Neural Network (CNN) that uses High Pass Filter (HPF) and extra-embedded data. SA-CNN is trained on a dataset created from two different datasets: Malimg dataset that is used to obtain malicious content and BOSSBase dataset that is used to obtain cover images. Concerning benign content, i.e., mere texts, they are selected randomly from the book [50] available at Project Gutenberg [9]. The obtained cover images, malicious, and benign contents are used to generate stego-images. These stego-images form our initial dataset. This dataset is strengthened by hiding additional content for increasing the hiding rate and evaluating the accuracy of SA-CNN to detect hidden content.

(b) The second CNN, called MBC-CNN is used for classifying the extracted hidden content, obtained thanks to SA-CNN, into malicious or benign content. Extensive experiments were conducted to evaluate the performance of SA-CNN powered with MBC-CNN**.** When compared with Simulated Dual-CNN [45], SA-CNN outperformances, in terms of accuracy.

As the perspectives of our work, a fixed time-based evaluation using a parallel approach or using Hadoop in the process of training can be paralyzed. In addition, more datasets can be involved in the evaluation process.

Acknowledgments

The authors are thankful to the Deanship of Graduate Studies and Scientific Research at University of Bisha for supporting this work through the Fast-Track Research Support Program.

Conflicts of Interest: "The authors declare no conflict of interest."

DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024

References

[1] Swetha, V., V. Prajith, and V. Kshema. "Data hiding using video steganography-a survey." International Journal of Science, Engineering and Computer Technology 5.6 (2015): 206.

[2] Taha, Mustafa Sabah, et al. "Combination of steganography and cryptography: A short survey." IOP conference series: materials science and engineering. Vol. 518. No. 5. IOP Publishing, 2019.

[3] Duluta, Andrei, et al. "Secure communication method based on encryption and steganography." 2017 21st International Conference on Control Systems and Computer Science (CSCS). IEEE, 2017.

[4] Douglas, Mandy, et al. "An overview of steganography techniques applied to the protection of biometric data." Multimedia Tools and Applications 77.13 (2018): 17333-17373.

[5] Fridrich, Jessica, Miroslav Goljan, and Dorin Hogea. "Steganalysis of JPEG images: Breaking the F5 algorithm." International Workshop on Information Hiding. Springer, Berlin, Heidelberg, 2002.

[6] Shankar, Deepa D., and Adresya Suresh Azhakath. "Minor blind feature based Steganalysis for calibrated JPEG images with cross validation and classification using SVM and SVM-PSO." Multimedia Tools and Applications 80.3 (2021): 4073-4092.

[7] BOSSWebsite,Online[Available]: http://agents.fel.cvut.cz/boss/index.php?mode=VIEW&tmpl=materials (Accessed 1 May 2021)

[8] Malware Classification on Malimg Dataset, online [Available]: https://paperswithcode.com/sota/malware-classification-on-malimg-dataset (Accessed 27 July 2021)

[9] Project Gutenberg, a library of free eBooks, November 13, 2004, online [Available]: https://www.gutenberg.org/ebooks/ (Accessed 27 July 2021).

[10] Fridrich, Jessica, and Miroslav Goljan. "Practical steganalysis of digital images: state of the art." security and Watermarking of Multimedia Contents IV. Vol. 4675. SPIE, 2002.

[11] Westfeld, Andreas, and Andreas Pfitzmann. "Attacks on steganographic systems." International workshop on information hiding. Springer, Berlin, Heidelberg, 1999.

[12] Fridrich, Jessica, Miroslav Goljan, and Rui Du. "Reliable detection of LSB steganography in color and grayscale images." Proceedings of the 2001 workshop on Multimedia and security: new challenges. 2001.

[13] Harmsen, Jeremiah Joseph, and William A. Pearlman. "Steganalysis of additive-noise modelable information hiding." Security and Watermarking of Multimedia Contents V. Vol. 5020. SPIE, 2003.

[14] Liu, Shaohui, Hongxun Yao, and Wen Gao. "Steganalysis of data hiding techniques in wavelet domain." International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004, Vol. 1. IEEE, 2004.

[15] Kim, Jaeyoung, Hanhoon Park, and Jong-Il Park. "CNN-based image steganalysis using additional data embedding." Multimedia Tools and Applications 79.1 (2020): 1355-1372.

[16] Xu, G., Xu, Y., Zhang, S., & Xie, X. (2021). SFRNet: Feature extraction-fusion Steganalysis network based on squeeze-and-Excitation block and RepVgg block. Security and Communication Networks, 2021, 1-11. https://doi.org/10.1155/2021/3676720

[17] Ge, Y., Zhang, T., Liang, H., Jiang, Q., & Wang, D. (2021). A novel technique for image Steganalysis based on separable convolution and adversarial mechanism. Electronics, 10(22), 2742. https://doi.org/10.3390/electronics1022274

[18] Reinel, T., Brayan, A. H., Alejandro, B. M., Alejandro, M., Daniel, A., Alejandro, A. J., Buenaventura, B. A., Simon, O., Gustavo, I., & Raul, R. (2021). GBRAS-net: A Convolutional neural network architecture for spatial image Steganalysis. IEEE Access, 9, 14340-14350. https://doi.org/10.1109/access.2021.3052494

[19] Lwowski, J., Corley, I., & Hoffman, J. (2020). Neural Steganalysis with spatial rich models for image steganography detection. https://doi.org/10.36227/techrxiv.11949762

[20] Singh, B., Sur, A., & Mitra, P. (2021). Steganalysis of digital images using deep fractal network. IEEE Transactions on Computational Social Systems, 8(3), 599-606. https://doi.org/10.1109/tcss.2021.3052520

DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024

[21] Pant, D., & Bista, R. (2021). Image-based malware classification using deep Convolutional neural network and transfer learning. 2021 3rd International Conference on Advanced Information Science and System (AISS 2021). https://doi.org/10.1145/3503047.3503081

[22] Alam, M., Akram, A., Saeed, T., & Arshad, S. (2021). DeepMalware: A deep learning based malware images classification. 2021 International Conference on Cyber Warfare and Security (ICCWS). https://doi.org/10.1109/iccws53234.2021.970302.

[23] Xu, Guanshuo. "Deep convolutional neural network to detect J-UNIWARD." Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security. 2017.

[24] Tang, Weixuan, et al. "Automatic steganographic distortion learning using a generative adversarial network." IEEE Signal Processing Letters 24.10 (2017): 1547-1551.

[25] Ye, Jian, Jiangqun Ni, and Yang Yi. "Deep learning hierarchical representations for image steganalysis." IEEE Transactions on Information Forensics and Security 12.11 (2017): 2545-2557.

[26] Yedroudj, Mehdi, Frédéric Comby, and Marc Chaumont. "Yedroudj-net: An efficient CNN for spatial steganalysis." 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018.

[27] Tsang, Clement Fuji, and Jessica Fridrich. "Steganalyzing images of arbitrary size with CNNs." Electronic Imaging 2018.7 (2018): 121-1.

[28] Zhang, Ru, et al. "Efficient feature learning and multi-size image steganalysis based on CNN." arXiv preprint arXiv:1807.11428 (2018).

[29] Wu, Songtao, Shenghua Zhong, and Yan Liu. "Deep residual learning for image steganalysis." Multimedia tools and applications 77.9 (2018): 10437-10453.

[30] Kim, Jaeyoung, Hanhoon Park, and Jong-Il Park. "CNN-based image steganalysis using additional data embedding." Multimedia Tools and Applications 79.1 (2020): 1355-1372.

[31] Chaganti, R., R, V., Alazab, M., & Pham, T. (2021). Stegomalware: A systematic survey of malware hiding and detection in images, machine learning models and research challenges. https://doi.org/10.36227/techrxiv.1675545

[32] Selvaraj, A., Ezhilarasan, A., Wellington, S. L., & Sam, A. R. (2020). Digital image steganalysis: A survey on paradigm shift from machine learning to deep learning based techniques. IET Image Processing, 15(2), 504-522. https://doi.org/10.1049/ipr2.12043

[33] Eid, W. M., Alotaibi, S. S., Alqahtani, H. M., & Saleh, S. Q. (2022). Digital image Steganalysis: Current methodologies and future challenges. IEEE Access, 10, 92321-92336. https://doi.org/10.1109/access.2022.3202905

[34] Shehab, D. A., & Alhaddad, M. J. (2022). Comprehensive survey of multimedia Steganalysis: Techniques, evaluations, and trends in future research. Symmetry, 14(1), 117. https://doi.org/10.3390/sym14010117

[35] Nataraj, Lakshmanan, et al. "Malware images: visualization and automatic classification." Proceedings of the 8th international symposium on visualization for cyber security. 2011.

[36] Hou, Xiaodan, et al. "Combating highly imbalanced steganalysis with small training samples using feature selection." Journal of Visual Communication and Image Representation 49 (2017): 243-256.

[37] Jia, Ju, et al. "An Effective Imbalanced JPEG Steganalysis Scheme based on Adaptive Cost-Sensitive Feature Learning." IEEE Transactions on Knowledge and Data Engineering (2020).

[38] Wu, Anxin, et al. "Unbalanced JPEG image steganalysis via multiview data match." Journal of visual communication and image representation 34 (2016): 103-107.

[39] Khormali, Aminollah, et al. "COPYCAT: practical adversarial attacks on visualization-based malware detection." arXiv preprint arXiv:1909.09735 (2019).

[40] Osunade, O., and I. A. Ganiyu. "Enhancing the least significant bit (LSB) algorithm for steganography." International Journal of Computer Applications 149.3 (2016): 1-8.

[41] Nagalla, S., & Inampudi, R. B. (2014). Perceptual Weights Based On Local Energy For Image Quality Assessment. International Journal of Image Processing (IJIP), 8(6), 468.

DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024

[42] Rouse, D. M., & Hemami, S. S. (2008, October). Understanding and simplifying the structural similarity metric. In 2008 15th IEEE international conference on image processing (pp. 1188-1191). IEEE.

[43] Rustad, S., Syukur, A., & Andono, P. N. (2021). Inverted LSB image steganography using adaptive pattern to improve imperceptibility. Journal of King Saud University-Computer and Information Sciences.

[44] Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC genomics, 21(1), 1-13.

[45] Kim, Jaeyoung, Hanhoon Park, and Jong-Il Park. "CNN-based image steganalysis using additional data embedding." Multimedia Tools and Applications 79.1 (2020): 1355-1372.

[46] Fridrich J, Kodovsky J (2011) Rich models for steganalysis of digital images. IEEE Transac-´ tions on Information Forensics and Security 7(3):868–882. http://dde.binghamton.edu/kodovsky/pdf/ TIFS2012-SRM.pdf

[47] Bas, P., Filler, T., Pevný, T. (2011). ”Break Our Steganographic System”: The Ins and Outs of Organizing BOSS. In: Filler, T., Pevný, T., Craver, S., Ker, A. (eds) Information Hiding. IH 2011. Lecture Notes in Computer Science, vol 6958. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-24178-9_5

[48] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath. 2011. Malware images: visualization and automatic classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security (VizSec '11). Association for Computing Machinery, New York, NY, USA, Article 4, 1–7. https://doi.org/10.1145/2016904.2016908

[49] Yuan Y, Lu W, Feng B, Weng J (2017) Steganalysis with CNN using multi-channels filtered residuals. ICCCS:110–120.

[50] Thomas Jefferson, "The Declaration of Independence of the United States of America", Prabhat Prakashan, 2021.

DOI: https://doi.org/10.54216/FPA.170228
Received: February 25, 2024 Revised: May 24, 2024 Accepted: October 28, 2024