Sensitivity of deep learning applied to spatial image steganalysis

Reinel Tabares-Soto [Corresp., 1] , **Harold Brayan Arteaga-Arteaga** [1] , **Alejandro Mora-Rubio** [1] , **Mario Alejandro Bravo-Ortíz** [1] , **Daniel Arias-Garzón** [1] , **Jesús Alejandro Alzate-Grisales** [1] , **Simon Orozco-Arias** [2, 3] , **Gustavo Isaza** [3] , **Raúl Ramos-Pollán** [4]

[1] Department of Electronics and Automation, Universidad Autónoma de Manizales, Manizales, Caldas, Colombia

[2] Department of Computer Science, Universidad Autónoma de Manizales, Manizales, Caldas, Colombia

[3] Department of Systems and Informatics, Universidad de Caldas, Manizales, Caldas, Colombia

[4] Department of Systems Engineering, Universidad de Antioquia, Medellín, Antioquia, Colombia

Corresponding Author: Reinel Tabares-Soto
Email address: rtabares@autonoma.edu.co

In recent years, the traditional approach to spatial image steganalysis has shifted to Deep Learning (DL) techniques, which have improved the detection accuracy while combining feature extraction and classification in a single model, usually, a Convolutional Neural Network (CNN). The main contribution from researchers in this area are new architectures that further improve the detection accuracy. Nevertheless, the preprocessing and partition of the database influence the overall performance of the CNN. This paper presents the results achieved by novel steganalysis networks (Xu-Net, Ye-Net, Yedroudj-Net, SR-Net, Zhu-Net, and GBRAS-Net) using different combinations of image and filter normalization ranges, various database splits, a diverse composition of the training mini-batches, different activation functions for the preprocessing stage, as well as an analysis on the activation maps and how to report accuracy. These results demonstrate how sensible steganalysis systems are to changes in any stage of the process and how important it is for researchers in this field to register and report their work thoroughly. We also propose a set of recommendations for the design of experiments in steganalysis with DL.

Sensitivity of deep learning applied to Spatial Image Steganalysis

Reinel Tabares-Soto[1], **Harold Brayan Arteaga-Arteaga**[1], **Alejandro Mora-Rubio**[1], **Mario Alejandro Bravo-Ortiz**[1], **Daniel Arias-Garzón**[1], **Jesus Alejandro Alzate-Grisales**[1], **Simon Orozco-Arias**[2,3], **Gustavo Isaza**[3], and **Raúl Ramos-Pollán**[4]

[1]Department of Electronics and Automation, Universidad Autónoma de Manizales, Manizales, Caldas, Colombia
[2]Department of Computer Science, Universidad Autónoma de Manizales, Manizales, Caldas, Colombia
[3]Department of Systems and Informatics, Universidad de Caldas, Manizales, Caldas, Colombia
[4]Department of Systems Engineering, Universidad de Antioquia, Medellín, Antioquia, Colombia

Corresponding author:
Reinel Tabares-Soto[1]

Email address: rtabares@autonoma.edu.co

ABSTRACT

In recent years, the traditional approach to spatial image steganalysis has shifted to Deep Learning (DL) techniques, which have improved the detection accuracy while combining feature extraction and classification in a single model, usually, a Convolutional Neural Network (CNN). The main contribution from researchers in this area are new architectures that further improve detection accuracy. Nevertheless, factors such as the preprocessing stage and partition of the database influence the overall performance of the CNN. This paper presents the results achieved by novel steganalysis networks (Xu-Net, Ye-Net, Yedroudj-Net, SR-Net, Zhu-Net, and GBRAS-Net) using different combinations of image, and filter normalization ranges, multiple databases splits, a diverse composition of the training mini-batches, a variety activation functions for the preprocessing stage, as well as an analysis on the activation maps and how to report accuracy. These results demonstrate how sensitive steganalysis systems are to changes in any stage of the process and how important it is for researchers in this field to record and report their work thoroughly. We also propose a set of recommendations for experiments' design in steganalysis using DL.

INTRODUCTION

In the context of criptography and information hiding, steganography refers to hiding messages in digital multimedia files (Hassaballah, 2020; Hassaballah et al., 2021) and steganalysis consists of detecting whether a file has a hidden message or not (Reinel et al., 2019; Tabares-Soto et al., 2020; Chaumont, 2020). In digital image steganography, a message can be hidden by changing the value of some pixels in the image (spatial domain, see **Fig. 1**) (Hameed et al., 2019) or by modifying the coefficients of a frequency transform (frequency domain) while remaining invisible to the human eye. Some of the steganographic algorithms in the spatial domain are HUGO (Pevny et al., 2010b), HILL (Li et al., 2014), MiPOD (Sedighi et al., 2016), S-UNIWARD (Holub et al., 2014), and WOW (Holub and Fridrich, 2012).

On the other end, there are two main stages in the steganalysis process. Stage one is feature extraction, the most prominent technique being Rich Models (RM) (Fridrich and Kodovsky, 2012), while stage two involves a binary classification model, such as Support Vector Machines (SVM) or perceptrons, to predict if an image is steganographic or not. Nowadays, thanks to the evolution of Deep Learning (DL) (Theodoridis, 2015) and improvements in computing hardware capabilities (e.g., Graphics Processing

46 Unit (GPU) (Tabares Soto, 2016) and Tensor Processing Unit (TPU)), unifying both stages under the same
47 model, usually a Convolutional Neural Network (CNN), has become the go-to strategy, improving the
48 detection accuracy of steganographic images. Before the introduction of CNN in steganalysis, the state-
49 of-the-art approaches were Spatial Rich Models (SRM) (Fridrich and Kodovsky, 2012) and a Subtractive
50 Pixel Adjacency Matrix (SPAM) (Pevny et al., 2010a).

Cover image - Stego image = Steganographic content

Cover image

Stego image

Steganographic content

Figure 1. **Cover, stego and steganographic content images.**

51 The first CNN architecture applied to steganalysis, with five convolutional layers, a Gaussian activation,
52 and trained using supervised learning, was proposed by Qian et al. (2015). The following year, Xu et al.
53 (2016) proposed a CNN with five convolutional layers. The introduction of an absolute value (ABS) layer
54 and 1×1 convolutional kernels improved the detection accuracy over Qian's et al. proposal. Despite an
55 improvement in the detection accuracy, these CNN still did not outperform SRM (Fridrich and Kodovsky,
56 2012) or SPAM (Pevny et al., 2010a).
57 It was not until Ye et al. (2017) presented a more prominent CNN architecture with eight con-
58 volutional layers, that CNN models improved the detection accuracy over the traditional approaches.
59 This network also introduced a novel activation function named Truncation Linear Unit (TLU), and an
60 image-preprocessing layer using filters initialized with SRM-based weights. This approach improved
61 the detection accuracy by approximately 10% compared to the traditional algorithms and previous CNN.
62 Aiming to join the useful elements from earlier proposals, Yedroudj et al. (2018) proposed a new CNN
63 consisting of: SRM-based filter banks, five convolutional layers, Batch Normalization (BN), and TLU
64 activation units. Furthermore, the researchers proposed an augmented training database, by adding images
65 from the BOWS 2 database (Mazurczyk and Wendzel, 2017) to the traditional BOSSBase database (Bas
66 et al., 2011), and by including operations such as cropping, resizing, rotation, and interpolation. In the
67 same year, a CNN able to detect steganographic images in the spatial and frequency domain was proposed
68 by Boroumand et al. (2019), the main feature of this architecture is the use of residual connections.
69 Following the most relevant proposals in the steganalysis field, the CNN presented by Zhang et al.
70 (2019) introduced separate convolutions and multi-level average pooling known as Spatial Pyramid
71 Pooling (SPP) (He et al., 2014), which allows the network to process arbitrarily sized images. Tan et al.
72 (2020) sought to decrease the computational cost, storage overheads, and difficulties in training and
73 deployment. The resulting model (i.e., CALPA-Net) improved adaptivity, transferability, and scalability.
74 Furthermore, Wang et al. (2020) proposed a CNN that uses detection mechanisms and joint domains. The
75 authors applied SRM filters and the discrete cosine transform residual (DCTR) patterns for transformation
76 steganographic impacts.
77 Currently, GBRAS-Net architecture, presented by Reinel et al. (2021), achieves the highest detection
78 percentages of steganographic images in the spatial domain. In the preprocessing stage, this network keeps
79 the 30 SRM filters and uses a modified TanH activation function. This CNN involves skip-connections,
80 separable and depthwise convolutions using the ELU activation function, for feature extraction. For
81 the classification stage, the CNN uses a softmax directly after global average pooling, removing fully
82 connected layers.
83 **Table 1** shows the performance of the CNN architectures. These results correspond to the most
84 relevant architectures for classifying S-UNIWARD and WOW steganographic images. The payloads used
85 are 0.2 and 0.4 bits per pixel (bpp).

Table 1. Accuracy percentage of models for S-UNIWARD and WOW steganographic algorithms, with payloads of 0.2 and 0.4 bpp. The bold entries indicate the best results.

Year - Algorithm	S-UNIWARD 0.2 bpp	S-UNIWARD 0.4 bpp	WOW 0.2 bpp	WOW 0.4 bpp
2020 - GBRAS-Net	**73.6**	**87.1**	**80.3**	**89.8**
2019 - Zhu-Net	71.4	84.5	76.9	88.1
2018 - SR-Net	67.7	81.3	75.5	86.4
2018 - Yedroudj-Net	63.5	77.4	72.3	85.1
2017 - Ye-Net	60.1	68.7	66.9	76.7
2016 - Xu-Net	60.9	72.7	67.5	79.3
2015 - Qian-Net	53.7	69.1	61.4	70.7
2012 - SRM+Ensemble classifier	63.4	75.3	63.5	74.5

In general, a sensitivity analysis refers to the assessment of how the output of a system, or in this case performance of a model, is influenced by its inputs (Razavi et al., 2021), not only training data, but model hyper-parameters, preprocessing operations, and desing choices as well. Besides assuring the quality of a model (Saltelli et al., 2019), sensitivity analysis can provide an important tool in reporting reproducible results, by explaining the conditions around which those results were achieved (Razavi et al., 2021). In its most simple form, consists of varying each of the inputs around its possible values and evaluating the results achieved.

Given the accelerated growth of DL techniques for steganalysis, measuring how factors such as image and filter normalization, database partition, the composition of training mini-batches, and activation function can affect the development and performance of algorithms for steganographic images detection is essential. This research was motivated by the lack of detailed documentation of the experimental set-up, the difficulty to reproduce the CNNs, and the variability of reported results. This paper describes the results of a thorough experimentation process in which different CNN architectures were tested under different scenarios to determine how the training conditions affect the results. Similarly, this paper presents an analysis of how researchers can select the products to report, aiming to deliver reproducible and consistent results. These issues are essential to assess the sensitivity of DL algorithms to different training settings and will ultimately contribute to a further understanding of the problems applied to steganalysis and how to approach them.

The paper has the following order: Section "**Materials and Methods**" describes the database, CNN architectures, experiments, training and hyper-parameters, hardware and resources. Section "**Results**" presents the quantitative results found for each of the scenarios. Section "**Discussion**" discusses the results presented in terms of their relationship and effect on steganalysis systems. Lastly, Section "**Conclusions**" presents the conclusions of the paper.

MATERIALS AND METHODS

Database
The database used for the experiments was *Break Our Steganographic System* (BOSSBase 1.01) (Bas et al., 2011). This database consists of 10,000 cover images of 512×512 pixels in a Portable Gray Map (PGM) format (8 bits grayscale). For this research, similar to the process presented by Tabares-Soto et al. (2021), the following operations were performed on the images:

- All images were resized to 256×256 pixels.

- Each corresponding steganographic image was created for each cover image using S-UNIWARD (Holub et al., 2014) and WOW (Holub and Fridrich, 2012) with payload 0.4 bpp. The implementation of these steganographic algorithms was based on the open-source tool named Aletheia (Lerch, 2020) and the Digital Data Embedding Laboratory at Binghamton University (University, 2015).

- The images were divided into training, validation, and test sets. The size of each group varied according to the experiment.

Default partition
After the corresponding steganographic images are generated, the BOSSBase 1.01 database contains 10,000 pairs of images (cover and stego) divided into 4,000 pairs for train, 1,000 pairs for validation, and 5,000 for the test. This partition of the BOSSBase 1.01 database was based on Xu et al. (2016); Ye et al. (2017); Yedroudj et al. (2018); Boroumand et al. (2019); Zhang et al. (2019); Reinel et al. (2021).

CNN Architectures

The CNN architectures used in this research, except for GBRAS-Net, were modified according to the strategy described in Tabares-Soto et al. (2021) to improve the performance of the networks regarding convergence, stability of the training process, and detection accuracy. The modifications involved the following: a preprocessing stage with 30 SRM filters and a modified TanH activation with range $[-3, 3]$, Spatial Dropout before the convolutional layers, Absolute Value followed by Batch Normalization after the convolutional layers, Leaky ReLU activation in convolutional layers, and a classification stage with three fully connected layers (Bravo Ortíz et al., 2021). **Figure 2** shows two of the six CNN architectures used for the experiments.

Complexity of CNNs

There are two dimensions to calculate the computational complexity of a CNN, spatial and temporal. The spatial complexity calculates the disk size that the model will occupy after being trained (parameters and feature maps). The time complexity allows calculating floating-point operations per second (FLOPS) that the network can perform (He and Sun, 2015). **Equation 1** is used to calculate the temporal complexity of a CNN and **Equation 2** is used to calculate the spatial complexity.

$$Time \sim O\left(\sum_{l=1}^{D} M_l^2 . K_l^2 . C_{l-1} . C_l\right) \tag{1}$$

$$Space \sim O\left(\sum_{l=1}^{D} K_l^2 . C_{l-1} . C_l + \sum_{l=1}^{D} M_l^2 . C_l\right) \tag{2}$$

Where:

D = number of convolutional network layers (depth)

l = convolutional layer where the convolution process is being performed

M_l = is the size of one side of the feature map in the $l-th$ convolutional layer

K_l = is the size of one side of the kernel applied on the $l-th$ convolutional layer

C_{l-1} = number of channels of each convolution kernel at the input of the $l-th$ convolutional layer

C_l = number of convolution kernels at the output of the $l-th$ convolutional layer

It is important to clarify that for spatial complexity, the first summation calculates the total size of the network parameters. The second summation calculates the size of the feature maps. In **Table 2**, the spatial and temporal complexities of the CNNs worked in this sensitivity analysis can be observed.

Table 2. Spatial and temporal complexity of the CNNs used to perform the steganalysis process.

CNN	Total number of parameters for training	Spatial complexity In MegaBytes	Temporal Complexity In GigaFLOPS
Xu-Net	87,830	0.45	2.14
Ye-Net	88,586	0.43	5.77
Yedroudj-Net	252,459	1.00	12.51
SR-Net	4,874,074	19.00	134.77
Zhu-Net	10,233,770	39.00	3.07
GBRAS-Net	166,598	0.80	5.92

Experiments

Image normalization

Image normalization is a typical operation in digital image processing that changes the ranges of the pixel values to match the operating region of the activation function. The most used bounds for CNN training are 0 to 255, when the values are integers, and 0 to 1 with floating-point values. The selection of this range affects performance and, depending on the application, one or the other is preferred. The following ranges were tested to demonstrate these effects:

Figure 2. Convolutional Neural Network Architectures. (A) Xu-Net. And (B) GBRAS-Net.

161   • $[0, 255]$: 8 bit integer.

162   • $[-12, 8]$: minimum and maximum values of the original SRM filters.

163   • $[0, 1]$: activation function operating region.

164   • $[-1, 1]$: activation function operating region.

165   • $[-0.5, 0.5]$: activation function operating region.

SRM filters normalization

As for image normalization, the SRM filter values shown in **Fig. 3** impact network performance. To evaluate the effect of different filter values, experiments were performed without normalization and with normalization by a factor of $1/12$, which caused filter values to be in the range $[-1, 2/3]$.

Figure 3. The 30 SRM filters values

CNN input

To speed up the learning process and avoid issues with GPU memory limitation, CNN optimization is performed over batches of images rather than the complete training set. This dataset division means that the distribution of image classes within each batch of images affects the learning process. Three different image input approaches were tested to demonstrate the effect of the amount of stego and cover images in a batch on the learning process and determine the best way to input the images to the network. The first one involved putting all of the cover images, followed by all the stego images; the second one alternated cover and stego images. The third one applied a random approach.

Database partition

Dividing the database into three sets is good practice for artificial intelligence applications: the training set to adjust network parameters, the validation set to change network hyper-parameters, and the test set to perform the final evaluation of the CNN performance. There is a default partition (see *"Default partition"*) which most researchers use in the field. As part of the experimentation process developed in this research, the CNN was tested using three additional database partitions as follow (amounts in image pairs):

- Train: $2,500$, Validation: $2,500$, and Test: $5,000$.

- Train: $4,000$, Validation: $3,000$, and Test: $3,000$.

- Train: $8,000$, Validation: $1,000$, and Test: $1,000$.

Activation function of the preprocessing stage

The preprocessing stage, which consisted of a convolutional layer with 30 SRM filters, involves an activation function that affects model performance on specific steganographic algorithms. As part of the experimentation process, four different activation functions were tested: $3 \times TanH$, $3 \times HardSigmoid$, $3 \times Sigmoid$ and $3 \times Softsign$.

Activation maps analysis

The output of a particular layer of a CNN is known as activation maps, indicating how well the architecture performs feature extraction. This paper presents the comparative analysis of the activation maps generated by a cover, a stego ,and a "cover-stego" image in a trained model. Furthermore, by comparing ,it is possible to see the differences between them.

Accuracy reporting in steganalysis

One of the characteristics of CNN training in steganalysis is the unstable accuracy and loss values between epochs, leading to highly variable results and training curves. Consequently, an abnormally high accuracy value can be achieved at a given time during the training process. Although it is correct to select the best accuracy under comparison, having more data allows a better understanding of the CNN. For example, in this paper, model accuracy was evaluated using the mean and standard deviation of the top five results from training, validation, and testing.

Training and hyper-parameters

The training batch size was set to 64 images for Xu-Net, Ye-Net, Yedroudj-Net, and 32 for SR-Net, Zhu-Net, GBRAS-Net. The number of training epochs needed to reach convergence is 100, except for Xu-Net that uses 150 epochs. The spatial dropout rate was 0.1 in all layers. Batch normalization had a momentum of 0.2, epsilon of 0.001, and renorm momentum of 0.4. The stochastic gradient descent optimizer momentum was 0.95, and the learning rate was initialized to 0.005. Except for GBRAS-Net, all layers used a glorot normal initializer and L2 regularization for weights and bias. For GBRAS-Net architecture, the training network uses Adam optimizer, which has the following configuration: the learning rate is 0.001, the beta 1 is 0.9, the beta 2 is 0.999, the decay is 0.0, and the epsilon is $1e-08$. Convolutional layers, except the first layer of preprocessing, use a kernel initializer called glorot uniform. CNN uses a categorical cross-entropy loss for the two classes. *The metric used is accuracy.* Batch Normalization is configured like the other CNNs. In the original network, the maximum absolute value normalizes the 30 high-pass SRM filters for each filter. The same padding is used on all layers. As shown in **Fig. 3**, the predictions performed in the last part of the architecture directly use a Softmax activation function.

Hardware and resources

As previously described in Tabares-Soto et al. (2021), the architectures and experiment implementations used Python 3.8.1 and TensorFlow (Abadi et al., 2015) 2.2.0 in a workstation running Ubuntu 20.04 LTS as an operating system. The computer runs a GeForce RTX 2080 Ti (11 GB), CUDA Version 11.0, an AMD Ryzen 9 3950X 16-Core Processor, and 128 GB of RAM. The remaining implementations used the Google Colaboratory platform in an environment with a Tesla P100 PCIe (16 GB) or TPUs, CUDA Version 10.1, and 25.51 GB RAM.

GPUs and TPUs enhance deep learning models. Accessing TPUs is done from Google Colaboratory. Once there, the models are adjusted to work with the TPUs. For example, in the GBRAS-Net model, on an 11GB Nvidia RTX 2080Ti GPU (local computer), one epoch takes 229 seconds, whereas with the TPU configured in Google Colaboratory, the epoch needs only 52 seconds. That is, it performs it more than three times faster. For the Ye-Net model, on a 16GB Tesla P100 GPU (Google Colaboratory), one epoch took 44 seconds approximately; whereas with the TPU, it only takes 12 seconds. This verifies that the use of TPU helps the experiments run more efficiently. It is important to note that in Google Colaboratory, we can open several notebooks and use different accounts. Which helps reduce experiment times to an unprecedented level. To achieve a correct training of the CNN, batch sizes must be chosen for each model according to the hardware accelerator employed.

RESULTS

Image normalization

Image normalization is a typical operation in digital image processing that affects the performance of CNN. Different types of normalization processes were performed on the images (cover and stego) of BOSSBase 1.01 with WOW 0.4 bpp. Training and validation were performed with Xu-Net, Ye-Net, Yedroudj-Net, SR-Net, Zhu-Net, and GBRAS-Net CNNs (see **Fig. 2** for Xu-Net and GBRAS-Net), with default data partition (see *"Default partition"*) and no SRM filters normalization. Furthermore, the

244 distribution of image classes within each image batch was done based on a random distribution of the
245 training images (i.e., random positions of cover and stego images), a usual distribution of the validation
246 images (i.e, inputting all the cover images first, then all the stego images), and a usual distribution of the
247 test images.

Table 3. Image normalization and best test accuracy for CNNs with WOW 0.4 bpp using BOSSBase 1.01. The bold entries indicate the best results.

Image normalization	Test accuracy Xu-Net [%]	Test accuracy Ye-Net [%]	Test accuracy Yedroudj-Net [%]	Test accuracy SR-Net [%]	Test accuracy Zhu-Net [%]	Test accuracy GBRAS-Net [%]
$[0, 255]$	**82.6**	**84.8**	**85.5**	**84.8**	84.2	**88.4**
$[-12, 8]$	78.7	81.6	81.5	83.6	**84.9**	86.5
$[0, 1]$	65.9	72.7	51.0	50.5	78.0	84.4
$[-1, 1]$	51.4	76.3	52.1	75.9	79.7	84.4
$[-0.5, 0.5]$	64.2	76.2	50.6	50.2	77.7	85.1

248   **Table 3** shows the best test accuracy results with different image normalizations in the convolutional
249 neural networks with WOW 0.4 bpp. **Figure 4**, under the title "Image Normalization" shows the
250 accuracy curves of SR-Net, Zhu-Net, and GBRAS-Net CNNs with WOW 0.4 bpp for different image
251 normalizations.

SRM Filters Normalization

253 The SRM filters have an impact on the performance of CNNs for steganalysis. Therefore, filter normaliza-
254 tion was performed by multiplying by $1/12$. In **Table 4**, each image normalization, distribution of classes
255 within each batch of images, and data partition were equal to *"Image normalization"*; additionally, SRM
256 filter normalization was done by multiplying by $1/12$.

Table 4. SRM filters and image normalization and best test accuracy for CNNs with WOW 0.4 bpp using BOSSBase 1.01. The bold entries indicate the best results.

Image normalization	Test accuracy Xu-Net [%]	Test accuracy Ye-Net [%]	Test accuracy Yedroudj-Net [%]	Test accuracy SR-Net [%]	Test accuracy Zhu-Net [%]	Test accuracy GBRAS-Net [%]
$[0, 255]$	**79.7**	**82.6**	**81.6**	**82.8**	**84.9**	**87.1**
$[-12, 8]$	50.8	75.9	52.2	50.4	78.9	83.4
$[0, 1]$	50.4	69.6	50.2	81.4	50.0	83.6
$[-1, 1]$	50.1	66.8	50.2	81.2	50.8	84.6
$[-0.5, 0.5]$	50.2	63.1	50.0	81.5	50.0	85.5

257   **Table 4** shows the best test accuracy result with a different image and filter normalization in the
258 CNNs with WOW 0.4 bpp. **Figure 5**, under the title "SRM Filters Normalization" shows the accuracy
259 curves for SR-Net, Zhu-Net, and GBRAS-Net CNNs with WOW 0.4 bpp and a different image and filter
260 normalization.

CNN input

262 Three CNN input distributions were applied and mentioned in *"CNN input"* experiment, namely usual
263 (i.e., inputting all the cover images first, followed by all the stego images), random (random positions of
264 cover and stego images), and ordered (alternating cover and stego images). The three cases demonstrate
265 that the distribution of classes within each batch of images affects the learning process. The following
266 experiment (see **Table 5**) was performed using Ye-Net for S-UNIWARD 0.4 bpp, image pixel values in
267 the range [0,255], (original pixel values), with no SRM filter normalization, and a default data partition
268 (see *"Default partition"*).

Database partition

270 In artificial intelligence, the databases are divided into training, validation, and testing. For steganalysis, a
271 default data partition is used (see *"Default partition"*). **Table 6** and **Table 7** show the best accuracy results,
272 mean accuracy and Standard Deviation (SD) of the best training model with different data partitions,
273 image pixel values in the range [0,255], no SRM filter normalization, and distribution of classes within
274 each batch of images based on a random distribution of the training images and usual distributions of the
275 validation and test images.

Image Normalization
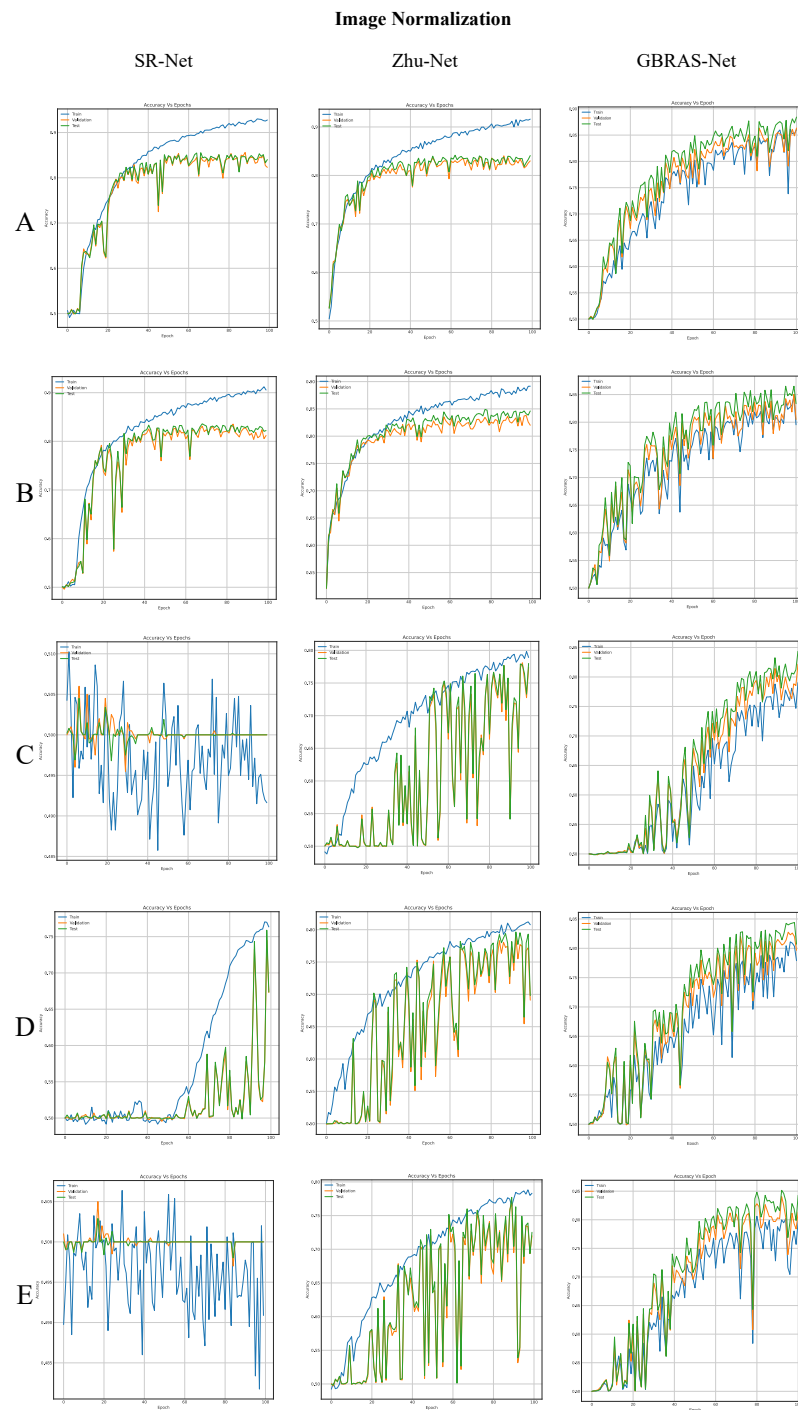
Figure 4. Accuracy curves for SR-Net, Zhu-Net and GBRAS-Net CNN with WOW 0.4 bpp and image normalization. (A) 0 to 255. (B) -12 to 8. (C) 0 to 1. (D) -1 to 1. (E) -0.5 to 0.5

276   **Table 6, and Fig. 6** shows the results of the different data partitions with S-UNIWARD 0.4 bpp.
277   **Table 7, and Fig. 7** shows the results of the different data partitions with WOW 0.4 bpp.

278   **Figures 8, 9, and 10** show the accuracy curves of SR-Net, Zhu-Net, and GBRAS-Net CNN with
279   S-UNIWARD and WOW 0.4 bpp for different data partitions.
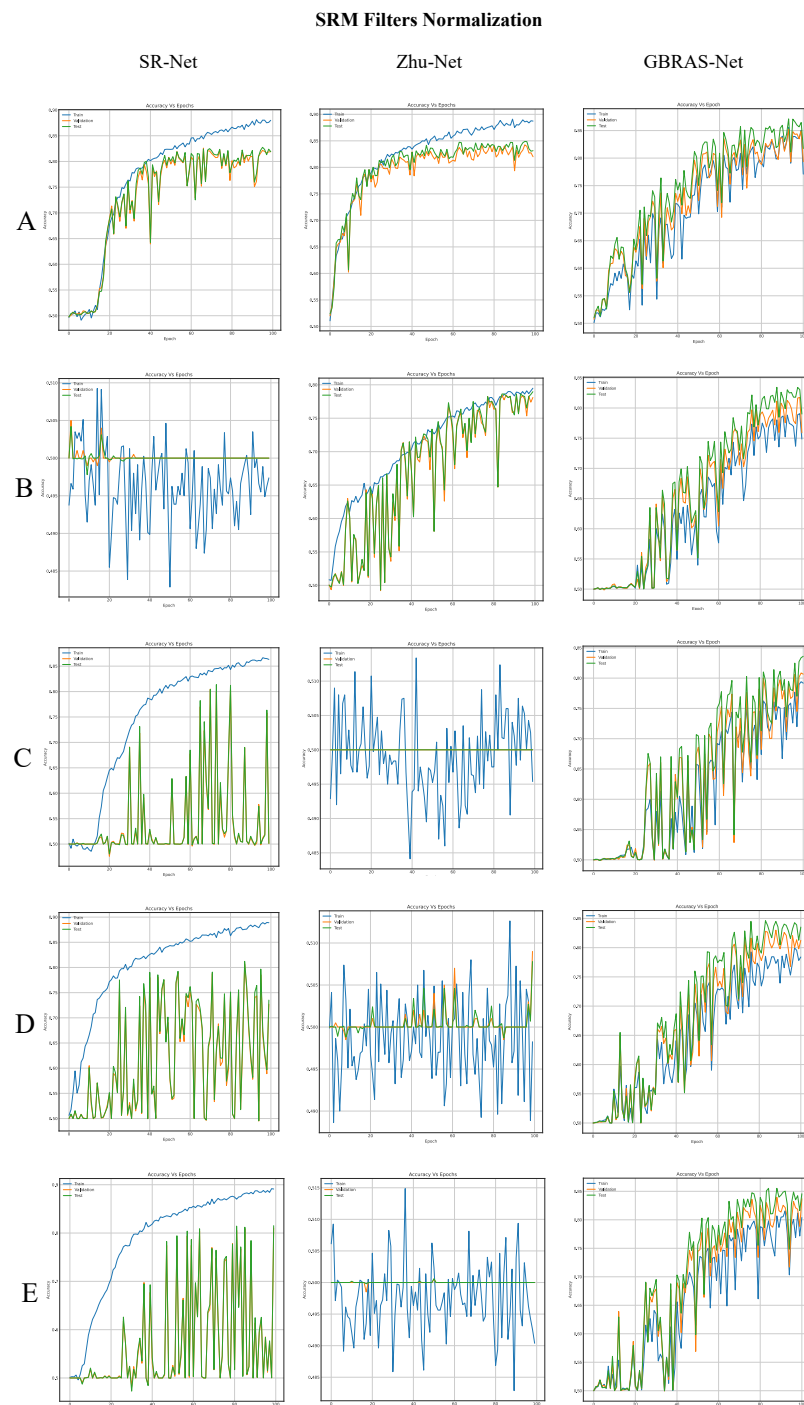
Figure 5. Accuracy curves for SR-Net, Zhu-Net and GBRAS-Net CNN with WOW 0.4 bpp and image with SRM filter normalization. (A) 0 to 255. (B) -12 to 8. (C) 0 to 1. (D) -1 to 1. (E) -0.5 to 0.5

Activation function of the preprocessing stage

Due to the sensitivity of the model, different modifications, such as changing the activation function, can generate variations in performance. The results achieved by Ye-Net with WOW and S-UNIWARD 0.4 bpp are shown in **Table 8**. The experiment was performed with a default data partition (see *"Default partition"*), image pixel values in the range [0,255], with no SRM filter normalization, and distribution of classes within each batch of images based on a random distribution of the training images and usual

Table 5. CNN input and best validation accuracy for Ye-Net with S-UNIWARD 0.4 bpp using BOSSBase 1.01.

Training image distribution	Validation image distribution	Best validation accuracy[%]
Random	Usual	83.4
Random	Random	83.9
Order	Order	84.1
Usual	Usual	84.3
Order	Usual	84.1
Order	Random	83.2
Random	Order	83.9
Usual	Random	83.8
Usual	Order	84.8

Table 6. Different data partitions and best test accuracy for CNNs with S-UNIWARD 0.4 bpp using BOSSBase 1.01. Corresponding to train, validation and test, respectively: (A) 4,000, 1,000, 5,000. (B) 2,500, 2,500, 5,000. (C) 4,000, 3,000, 3,000. And (D) 8,000, 1,000, 1,000.

CNN	Distribution	Accuracy on test [%]			CNN	Distribution	Accuracy on test [%]		
		Best	Mean	SD			Best	Mean	SD
Xu-Net	A	79.7	79.0	0.51	SR-Net	A	77.0	76.5	0.32
	B	78.4	77.3	1.01		B	73.3	73.1	0.23
	C	79.6	79.4	0.17		C	77.7	77.5	0.20
	D	85.0	84.4	0.33		D	87.5	87.4	0.14
Ye-Net	A	81.1	80.5	0.53	Zhu-Net	A	82.6	82.5	0.09
	B	77.2	76.8	0.41		B	81.2	80.5	0.35
	C	81.2	80.9	0.21		C	81.2	80.7	0.34
	D	86.8	86.0	0.63		D	86.9	86.7	0.13
Yedroudj-Net	A	81.8	81.1	0.47	GBRAS-Net	A	82.8	82.1	0.59
	B	78.5	77.5	0.91		B	80.8	79.5	1.19
	C	80.7	80.1	0.51		C	81.7	81.5	0.15
	D	86.3	85.5	0.63		D	89.1	88.3	0.45

Table 7. Different data partitions and best test accuracy for CNNs with WOW 0.4 bpp using BOSSBase 1.01. Corresponding to train, validation and test, respectively: (A) 4,000, 1,000, 5,000. (B) 2,500, 2,500, 5,000. (C) 4,000, 3,000, 3,000. And (D) 8,000, 1,000, 1,000.

CNN	Distribution	Accuracy on test [%]			CNN	Distribution	Accuracy on test [%]		
		Best	Mean	SD			Best	Mean	SD
Xu-Net	A	82.6	82.2	0.31	SR-Net	A	52.5	50.7	0.87
	B	81.4	81.1	0.22		B	84.2	83.5	0.53
	C	82.8	82.1	0.43		C	84.6	84.4	0.19
	D	87.3	86.8	0.21		D	89.4	89.1	0.13
Ye-Net	A	84.8	84.5	0.25	Zhu-Net	A	86.9	86.2	0.32
	B	82.7	82.2	0.37		B	83.8	83.6	0.16
	C	83.9	83.3	0.63		C	85.1	84.7	0.24
	D	88.1	87.7	0.27		D	88.4	88.2	0.14
Yedroudj-Net	A	85.5	85.1	0.33	GBRAS-Net	A	88.4	87.9	0.34
	B	84.1	83.5	0.35		B	87.0	86.5	0.35
	C	85.1	84.6	0.27		C	86.3	86.0	0.24
	D	88.7	88.4	0.15		D	89.4	89.2	0.13

distributions of the validation and test images.

Another important experiment that we can present is what happens when the value that multiplies the preprocessing activation function is changed. For WOW multiplying by 5, 8, 13, 21 the best results are: 82.9%, 83.0%, 82.3%, and 83.6% respectively. For S-UNIWARD multiplying by with 5, 8, 13, 21 the best results are: 85.0%, 84.3%, 85.1%, and 84.8% respectively.
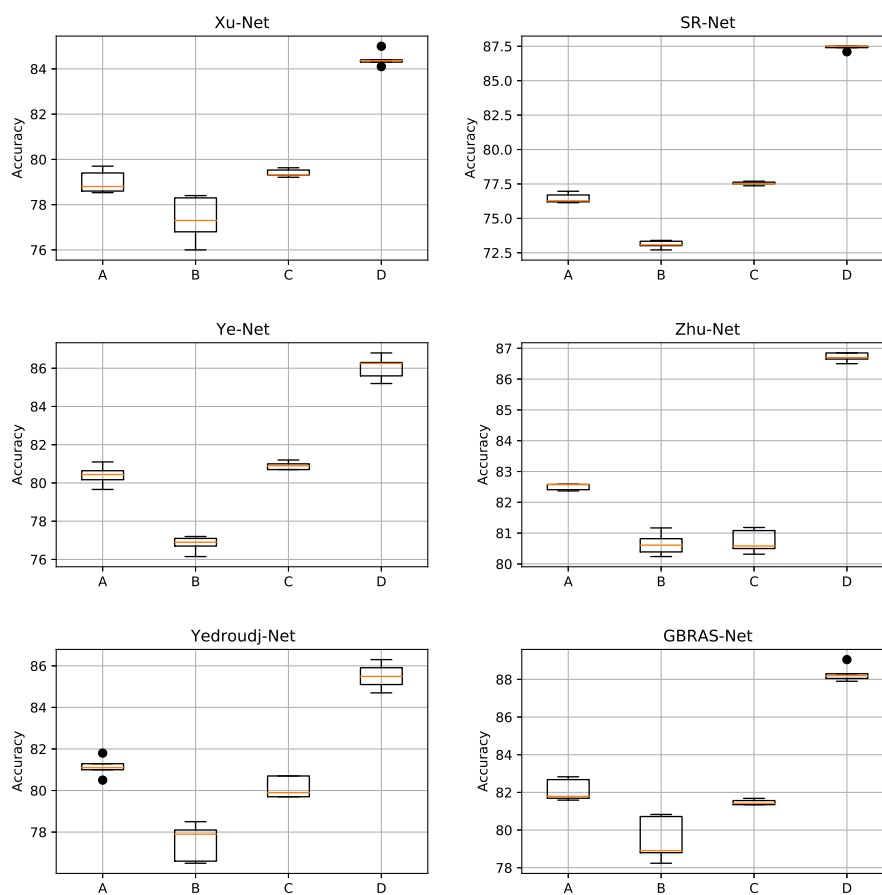
Figure 6. Boxplots for the S-UNIWARD experiments. This figure shows different data partitions experiments for novel CNN architectures. Train, Validation, Test: (A) 4,000, 1,000, 5,000. (B) 2,500, 2,500, 5,000. (C) 4,000, 3,000, 3,000. (D) 8,000, 1,000, 1,000.

Table 8. Effect of the activation function on two steganographic algorithms (WOW and S-UNIWARD) using Ye-Net architecture, trained on TPU with 200 epochs and batchsize of 64.

Activation function	WOW (Epoch) Accuracy	S-UNIWARD (Epoch) Accuracy
$3 \times TanH$	(119) 85.0	(196) 83.3
$3 \times HardSigmoid$	(162) 86.0	(188) 81.8
$3 \times Sigmoid$	(170) 85.5	(198) 81.8
$3 \times Softsign$	(154) 85.5	(163) 81.2

Activation maps for cover, stego, and steganographic noise images

The trained model has an accuracy of 89.8%, with BOSSBase 1.01. The model was implemented with GBRAS-Net and WOW 0.4 bpp, a default data partition (see *"Default partition"*), image pixel values in the range [0, 255], individual SRM filter normalization, and a class distribution within each batch of images based on a random distribution of the training images and usual distributions of the validation and test images. The activation maps of the first and the three last convolution of the network are shown in **Fig. 11**. The activation maps correspond to cover, stego, and steganographic noise images.

Figure 12 shows the ROC curves with Confidence Interval (CI) for the WOW steganography algorithm. BOSSBase 1.01 database was used to train the model. These curves correspond to the model presented in **Table 1** for GBRAS-Net. The ROC curves show the relationship between the false positive and true positive rates. These curves show the Area Under Curve (AUC) values; higher values indicate that the images were better classified by the computational model, which, in turn, depends on the steganography algorithm and payload.
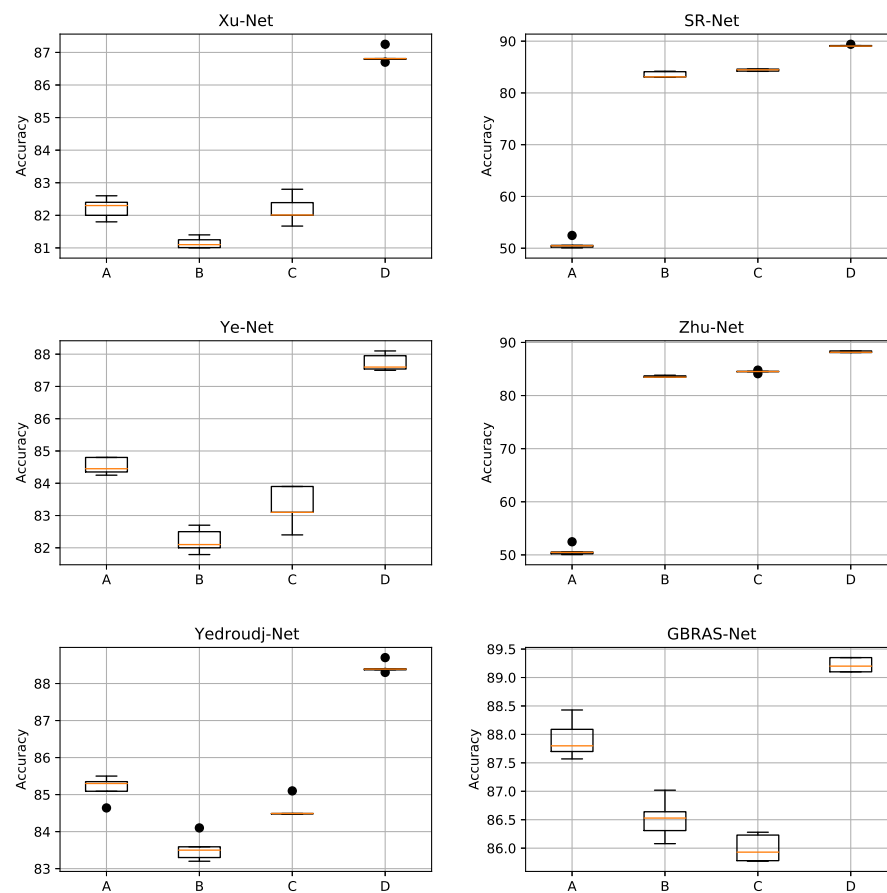
Figure 7. Boxplots for the WOW experiments. This figure shows different data partitions experiments for novel CNN architectures. Train, Validation, Test: (A) 4,000, 1,000, 5,000. (B) 2,500, 2,500, 5,000. (C) 4,000, 3,000, 3,000. (D) 8,000, 1,000, 1,000.

Accuracy reporting in steganalysis

The results of the experiment are shown with a data distribution consisting of $8,000$, $1,000$, and $1,000$ pairs of images, analyzed in GBRAS-Net and Xu-Net architecture using BOSSBase 1.01, image pixel values in the range [0,255], with no SRM filter normalization, and distribution of classes within each batch of images based on a random distribution of the training images and usual distributions of the validation and test images. **Table 9** shows the results of accuracy reporting. The model accuracy was evaluated using the mean and standard deviation of the top five results achieved by the CNN during training, validation, and testing.

DISCUSSION

This study presents results obtained from testing different combinations of image and filter normalization ranges, various database partitions, a diverse composition of training mini-batches, different activation functions for the preprocessing stage, as well as analysis on activation maps of convolutions and how to report accuracy when training six CNN architectures applied to image steganalysis in the spatial domain. The experiments proposed here show highly variable results, indicating the importance of detailed documentation and reports derived from novel work in this field.

Regarding image and SRM filter normalization, as shown in **Table 3**, the effectiveness of a normalization range depends on the selected CNN, such that SRM normalization (see **Table 4**) can generate completely different results.

The image normalization experiment demonstrates essential aspects of this analysis. For example, considering the Xu-Net architecture in **Table 3**, the best result is obtained using images with the original
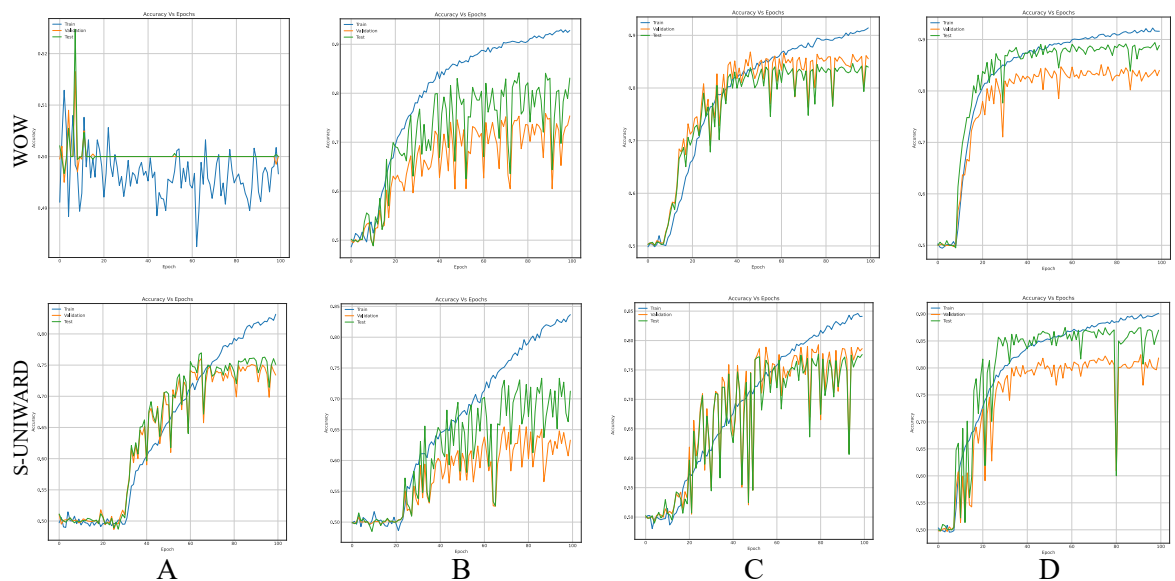
Figure 8. Accuracy curves of SR-Net with S-UNIWARD and WOW 0.4 bpp. This figure shows different data partitions for each row. Train, Validation, Test: (A) 4,000, 1,000, 5,000. (B) 2,500, 2,500, 5,000. (C) 4,000, 3,000, 3,000. (D) 8,000, 1,000, 1,000.
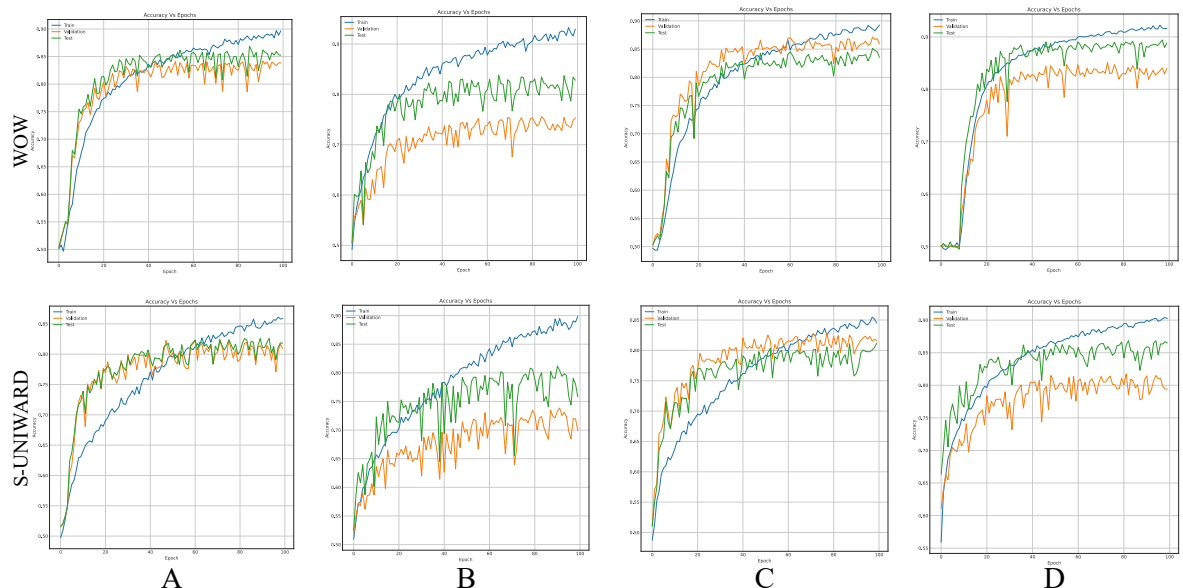
Figure 9. Accuracy curves of Zhu-Net with S-UNIWARD and WOW 0.4 bpp. This figure shows different data partitions for each row. Train, Validation, Test: (A) 4,000, 1,000, 5,000. (B) 2,500, 2,500, 5,000. (C) 4,000, 3,000, 3,000. (D) 8,000, 1,000, 1,000.

324 values of the database (i.e., in the range 0 to 255). Given this, one could conclude that there is no need
325 for image normalization in any architecture; however, a different result is observed with the Zhu-Net
326 architecture. Zhu-Net has the best result using the normalization of the pixels from -12 to 8 (inspired by
327 the minimum and maximum values of the original SRM filters). We recommend using the original pixel
328 values as the first option because it is the best option for most of CNN.
329   When considering the combination of image normalization and filter normalization, the results can be
330 different. For example, for SR-Net architecture from **Table 3**, the normalization of the pixels between
331 -0.5 to 0.5 generates an accuracy of only 50.2% without filter normalization. Conversely, with normalized
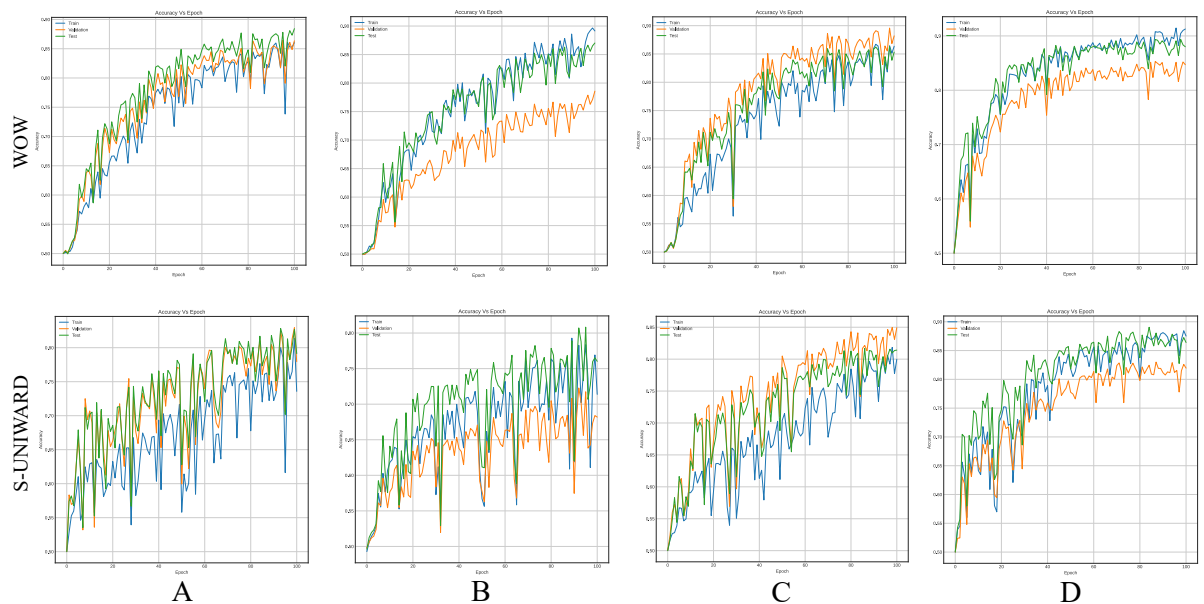
Figure 10. Accuracy curves of GBRAS-Net with S-UNIWARD and WOW 0.4 bpp. This figure shows different data partitions for each row. Train, Validation, Test: (A) 4,000, 1,000, 5,000. (B) 2,500, 2,500, 5,000. (C) 4,000, 3,000, 3,000. (D) 8,000, 1,000, 1,000.

Table 9. Accuracy report structure. Show the results with this manner allows understand how the model have the behavior for a specific experiment.

Xu-Net: Train=8,000, Valid=1,000, Test=1,000				GBRAS-Net: Train=8,000, Valid=1,000, Test=1,000			
S-UNIWARD 0.4bpp Best 5% Accuracies				S-UNIWARD 0.4bpp, the best 5% Accuracies			
Train	**Epoch**	**Valid**	**Test**	**Train**	**Epoch**	**Valid**	**Test**
83.4	149	77.6	83.3	88.4	99	82.6	87.3
83.1	144	77.0	85.0	88.1	90	81.2	86.5
82.9	145	77.4	83.5	87.9	96	81.6	86.8
82.7	147	76.7	83.4	87.9	86	80.8	85.8
82.7	148	77.3	83.1	87.8	87	83.0	88.1
82.9	*mean*	*77.2*	*83.6*	*88.0*	*mean*	*81.8*	*86.9*
0.32	*standard deviation*	*0.35*	*0.77*	*0.25*	*standard deviation*	*0.94*	*0.85*
Valid	**Epoch**	**Test**	**Train**	**Valid**	**Epoch**	**Test**	**Train**
77.6	143	83.9	81.6	83.0	87	88.1	87.8
77.6	149	83.3	83.4	82.9	94	87.9	87.6
77.4	145	83.5	82.9	82.9	71	88.3	86.5
77.4	150	84.1	82.2	82.8	77	88.2	85.8
77.3	130	84.0	82.3	82.6	99	87.3	88.4
77.5	*mean*	*83.7*	*82.5*	*82.8*	*mean*	*87.9*	*87.2*
0.12	*standard deviation*	*0.36*	*0.70*	*0.15*	*standard deviation*	*0.42*	*1.06*
Test	**Epoch**	**Train**	**Valid**	**Test**	**Epoch**	**Train**	**Valid**
85.0	144	83.1	77.0	89.1	84	87.2	82.5
84.4	141	82.7	76.7	88.3	71	86.5	82.9
84.4	131	82.2	77.3	88.2	77	85.8	82.8
84.3	133	81.0	76.3	88.1	87	87.8	83.0
84.2	140	82.2	77.1	87.9	76	82.7	82.2
84.4	*mean*	*82.2*	*76.9*	*88.3*	*mean*	*86.0*	*82.7*
0.33	*standard deviation*	*0.79*	*0.39*	*0.45*	*standard deviation*	*1.98*	*0.34*

SRMs, as shown in **Table 4**, the SR-Net CNN reaches an accuracy of up to 81.5%. However, as the normalization experiments show, GBRAS-Net is the architecture that best behaves or adapts to changes in data normalizations and distributions. We recommend making use of this new architecture.

Regarding the variability of accuracies, **Table 5** shows the results for the CNN input experiment and
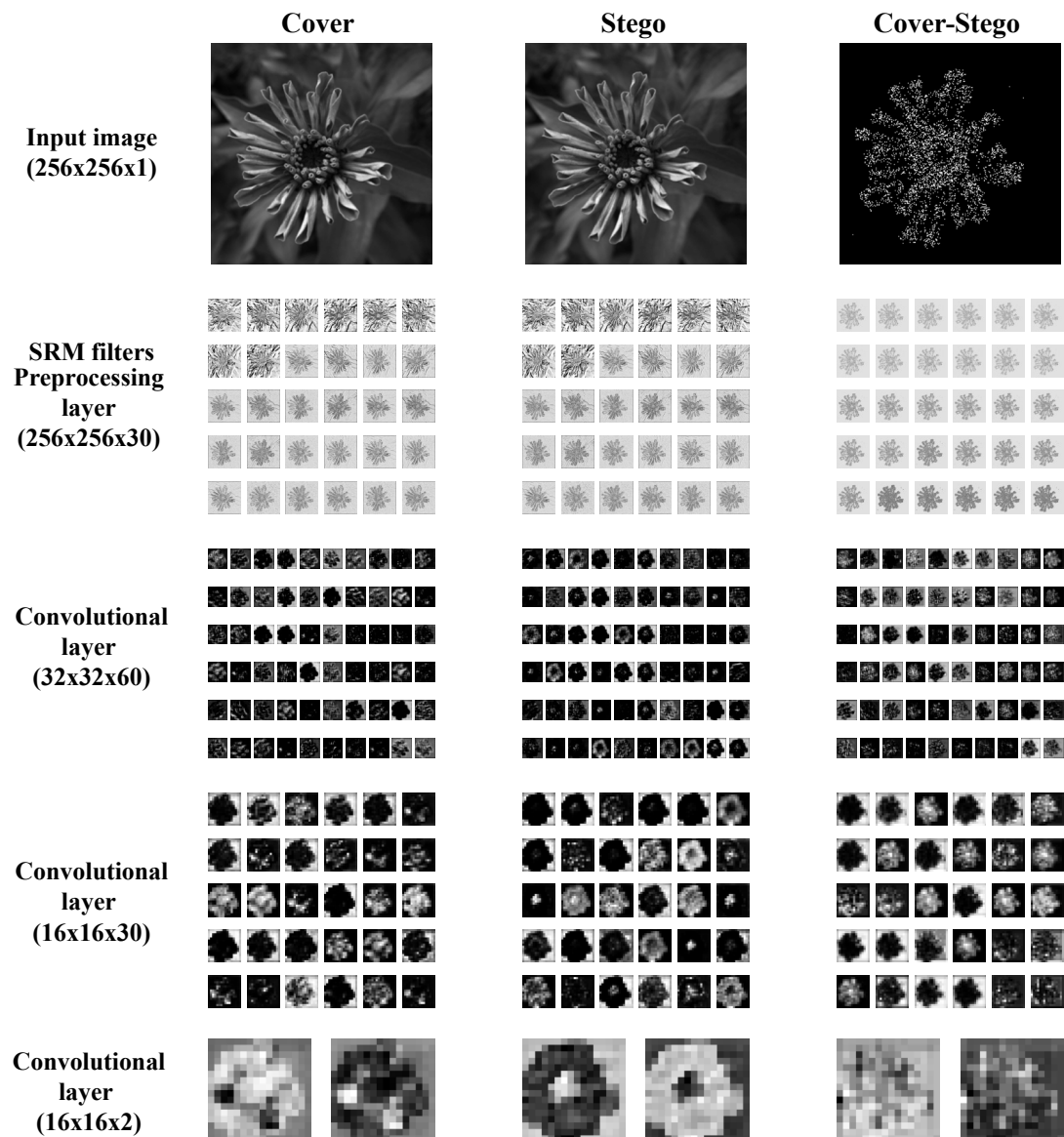
Figure 11. Activation maps of convolutional layers in GBRAS-Net architecture trained with WOW 0.4 bpp. This figure shows the Input image, the first convolutional layer or preprocessing layer with SRM Filters, and the last three convolutional layers.

provides a great example of the sensitivity of CNN for steganalysis. Based on these results, it is important to consider different image distribution options to train the networks if the objective is to present a network that achieves the highest possible accuracy.

In the database partition experiment, the architectures' detection accuracy improved as the training set increased and the test set decreased. Furthermore, if the test dataset reduces considerably, performance on future cases can be affected. In response, recent investigations use the BOWS 2 dataset since it contains more information and. Consequently, with a bigger dataset, data partition can have more information on training and test that can enhance performance. A small test set may be an inadequate representation of the distribution of the images that the network must classify in a production setting; thus, a higher detection accuracy with this partition may not lead to a helpful improvement.

Figures 8, 9, and 10 show that a smaller training set produces highly variable validation and test curves, while a bigger training set generates smoother curves. Furthermore, these curves show how the validation curve can sometimes be higher or lower than the training curve. For this reason, it is better to
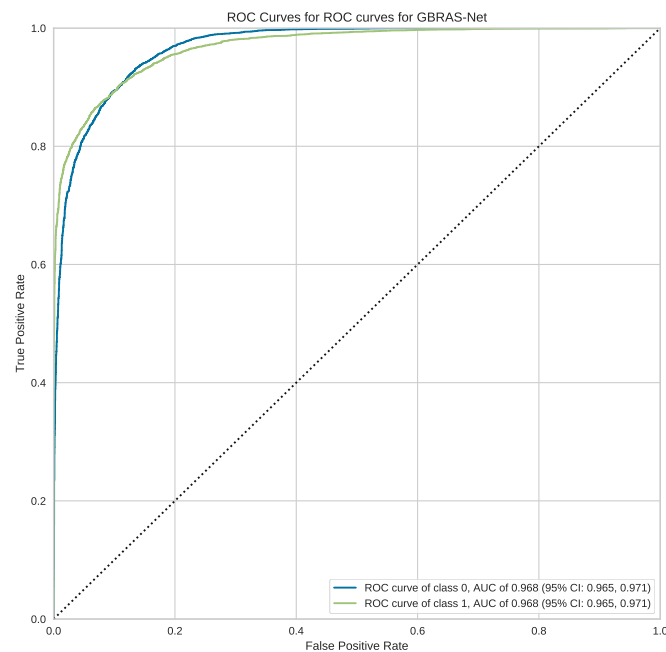
Figure 12. *ROC curves with CI for GBRAS-Net against WOW steganographic algorithm with* 0.4 *bpp on BOSSBase 1.01.*

choose the models from the results obtained in test data. For this reason, a good representation or quantity of test data is also important.

Table 8 shows that using different activation functions implies changes in performance. In Ye-Net for WOW and S-UNIWARD with $3 \times TanH$, an average accuracy of 84.2% is achieved, and with $3 \times HardSigmoid$, an average accuracy of 83.9%. Although for WOW, the best result is given by using the $3 \times HardSigmoid$ activation function overall. A model that serves for detection in several steganographic algorithms is better to use $3 \times TanH$ shown by the average value of accuracy.

Figure 11 shows that the activation maps from the stego image have differences with the cover image, which indicates a higher activation of the convolutional layer in the presence of the steganographic noise. Moreover, by comparing the activation maps, it is clear that a good learning process was achieved by extracting relevant features and focusing on borders and texture changes in the images, where the steganographic algorithms are known to embed most of the information. The analysis of the activation maps is an effective tool for researchers to evaluate the learning process and gain an understanding of the features that the CNN recognizes as relevant for the steganalysis task. This shows that GBRAS-Net has an excellent ability to discriminate between images without hidden content and with hidden content.

The design of CNN networks allows capturing steganographic content. The first layer (preprocessing), which contains the filters, is responsible for enhancing this noise while decreasing the content of the input image (See **Fig 11**. in the Cover and Stego columns for the SRM filters row). The Cover-Stego column in **Fig. 11** shows the noise. Adaptive steganography does its job well in adapting to image content; as seen in the image, it does so at hard-to-detect edges and places.

As proposed here **Table 9**), the main advantage of accuracy reporting is to be able to determine the consistency of the results based not only on the final value or the best one. To obtain these results, as the architectures are trained, a model is saved from each epoch. With these models, the accuracies are then obtained in the datasets. With this, you can know which are the best models. And with this accuracy reporting mode, when a specific experiment is presented, whoever will reproduce it will see the range of results to expect. As shown here, the sensitivity of deep learning is excellent in this problem, which can lead to reproducing a CNN not obtaining the same result from the reporter.

With all information shown in this work for spatial image steganalysis using deep learning, we propose a set of recommendations for the design of experiments, listed below:

- Recommendation 1: measure CNN sensitivity to data and SRM filter normalizations.

- Recommendation 2: measure CNN sensitivity to data distributions.

- Recommendation 3: measure CNN sensitivity to data splits.

- Recommendation 4: measure CNN sensitivity to activation functions on preprocessing stage.

- Recommendation 5: show activation maps of cover, stego, and steganographic noise images.

- Recommendation 6: report the top five best epochs with accuracies and their standard deviation.

Finally, the contributions of this paper will be listed at a general level:

- Sensitivity in the percentages of accuracy in detecting steganographic images when applying different normalizations in the pixels of the images on six architectures of CNNs (See **Table 3** and **Figures 4**).

- Sensitivity in the percentage of accuracy detecting steganographic images when applying different normalizations in the SRM filters in the preprocessing stage on six CNNs architectures (See **Table 4** and **Figures 5**).

- Sensitivity in the percentages of accuracy detecting steganographic images when feeding the CNNs with different distribution orders of the dataset in their training process (See **Table 5**).

- Sensitivity in the percentages of accuracy detecting steganographic images has the partition of the set of images in training, validation, and test (See **Table 6 and 7** and **Figures 8,9, and 10**).

- Sensitivity in the percentages of accuracy detecting steganographic images that have tested different activation functions in the preprocessing stage for the training process (See **Table 8**).

- The importance of analyzing the activation maps of the different convolutional layers to make new designs of CNNs architectures and understand their behavior (See **Figure 11**).

- The importance of reporting the average and standard deviation in the percentages of accuracy detecting steganographic images to determine the results reported in the experiments (See **Table 9**).

Some possible limitations of the current work, which was developed under the clairvoyant scenario, come from the nature and characteristics of the database: the use of images with fixed resolutions, the specific cameras used to take the pictures, the bit depth of the images, and that all the experiments were performed in the spatial domain. As future work, it is proposed to study each state-of-the-art CNNs weaknesses and problems to design new architectures and computational elements for steganalysis, taking papers (Vasan et al., 2020) and (Bhattacharya et al., 2021) as a reference.

CONCLUSIONS

As shown by the results presented in this paper, steganalysis detection systems are susceptible to changes in any stage of the process. Factors such as image and filter normalization ranges, database partition, the composition of training mini-batches, and activation function in the preprocessing stage affect the CNN performance to the point they determine its success. With this in mind, we present the analysis of the activation maps of convolutions for GBRAS-Net as a valuable tool to assess the CNN training process and its ability to extract distinctive features between cover and stego images. Understanding the behavior of steganalysis systems is key to design strategies and computational elements to overcome their limitations and improve their performance. For example, taking Ye-Net as a reference, using the WOW steganographic algorithm with 0.4 bpp, on the BOSSBase 1.01 database and the values of each pixel without any modification (0 and 255), results in an accuracy of 84.8% in the detection of steganographic images, while applying a normalization of the image pixels between 0 and 1 generates a result of 72.7% (See **Table 3**), taking into account the normal values of the SRM filters (-12 and 8), now if we normalize the values of the previous filters between 0 and 1 with the same characteristics mentioned above, we obtain results of 82.6% and 69.6% respectively (See **Table 4**). Now with the same CNN and performing different

422 partitions of the data set (training, validation, and test), we observe accuracy results on average between
423 76.8% and 86.0% for the S-UNIWARD steganographic algorithm with 0.4 bpp (See **Table 6**). The above
424 and the other results mentioned in this paper highlight the importance of clearly and precisely defining the
425 experiments performed in steganalysis to report the results reliably and facilitate the reproduction of the
426 experiments by the researchers.
427   Furthermore, we recommend reporting accuracy values as the mean and standard deviation of the top
428 five results, as it helps account for model consistency and reliability. If possible, we encourage researchers
429 to liberate a repository with code and data resources to reproduce the results and report the implementation
430 details thoroughly, taking into account preprocessing and feature extraction techniques, classification
431 process, and hyper-parameters.

DATA AVAILABILITY

433 All resources, including source code and databases of this project, are available as open-source software in
434 the following repository: https://github.com/BioAITeam/Sensitivity-of-deep-learning-applied-to-Spatial-
435 Image-Steganalysis

REFERENCES

437 Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J.,
438   Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser,
439   L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M.,
440   Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas,
441   F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). Tensorflow:
442   Large-scale machine learning on heterogeneous distributed systems.
443 Bas, P., Filler, T., and Pevny, T. (2011). "Break Our Steganographic System": The Ins and Outs of
444   Organizing BOSS. In *INFORMATION HIDING*, volume 6958/2011 of *Lecture Notes in Computer
445   Science*, pages 59–70, Czech Republic.
446 Bhattacharya, S., Reddy Maddikunta, P. K., Pham, Q.-V., Gadekallu, T. R., Krishnan S, S. R., Chowdhary,
447   C. L., Alazab, M., and Jalil Piran, M. (2021). Deep learning and medical image processing for
448   coronavirus (covid-19) pandemic: A survey. *Sustainable Cities and Society*, 65:102589.
449 Boroumand, M., Chen, M., and Fridrich, J. (2019). Deep Residual Network for Steganalysis of Digital
450   Images. *IEEE Transactions on Information Forensics and Security*, 14(5):1181–1193.
451 Bravo Ortíz, M. A., Arteaga Arteaga, H. B., Tabares Soto, R., Padilla Buriticá, J. I., and Orozco-Arias, S.
452   (2021). Cervical cancer classification using convolutional neural networks, transfer learning and data
453   augmentation. *Revista EIA*, 18(35):1–12.
454 Chaumont, M. (2020). 14 - deep learning in steganography and steganalysis. In Hassaballah, M., editor,
455   *Digital Media Steganography*, pages 321–349. Academic Press.
456 Fridrich, J. and Kodovsky, J. (2012). Rich Models for Steganalysis of Digital Images. *IEEE Transactions
457   on Information Forensics and Security*, 7(3):868–882.
458 Hameed, M. A., Hassaballah, M., Aly, S., and Awad, A. I. (2019). An adaptive image steganography
459   method based on histogram of oriented gradient and pvd-lsb techniques. *IEEE Access*, 7:185189–
460   185204.
461 Hassaballah, M. (2020). *Digital media steganography: principles, algorithms, and advances*. Academic
462   Press.
463 Hassaballah, M., Hameed, M. A., Awad, A. I., and Muhammad, K. (2021). A novel image steganography
464   method for industrial internet of things security. *IEEE Transactions on Industrial Informatics*, pages
465   1–1.
466 He, K. and Sun, J. (2015). Convolutional neural networks at constrained time cost. In *2015 IEEE
467   Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5353–5360.
468 He, K., Zhang, X., Ren, S., and Sun, J. (2014). Spatial Pyramid Pooling in Deep Convolutional Networks
469   for Visual Recognition. *CoRR*, abs/1406.4:1–14.
470 Holub, V. and Fridrich, J. (2012). Designing steganographic distortion using directional filters. In *2012
471   IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 234–239.
472 Holub, V., Fridrich, J., and Denemark, T. (2014). Universal distortion function for steganography in an
473   arbitrary domain. *EURASIP Journal on Information Security*, 2014(1):1.

474 Lerch, D. (2020). Aletheia. https://daniellerch.me/. 2020-10-10.

475 Li, B., Wang, M., Huang, J., and Li, X. (2014). A new cost function for spatial image steganography. In
476 *2014 IEEE International Conference on Image Processing (ICIP)*, pages 4206–4210.

477 Mazurczyk, W. and Wendzel, S. (2017). Information Hiding: Challenges for Forensic Experts. *Commun.*
478 *ACM*, 61(1):86–94.

479 Pevny, T., Bas, P., and Fridrich, J. (2010a). Steganalysis by Subtractive Pixel Adjacency Matrix. *IEEE*
480 *Transactions on Information Forensics and Security*, 5(2):215–224.

481 Pevny, T., Filler, T., and Bas, P. (2010b). Using High-Dimensional Image Models to Perform Highly
482 Undetectable Steganography. In Bohme, R., Fong, P. W. L., and Safavi-Naini, R., editors, *Information*
483 *Hiding*, pages 161–177, Berlin, Heidelberg. Springer Berlin Heidelberg.

484 Qian, Y., Dong, J., Wang, W., and Tan, T. (2015). Deep learning for steganalysis via convolutional neural
485 networks. *IS&T International Symposium on Electronic Imaging (EI 2015)*, 9409:94090J.

486 Razavi, S., Jakeman, A., Saltelli, A., Prieur, C., Iooss, B., Borgonovo, E., Plischke, E., Lo Piano, S.,
487 Iwanaga, T., Becker, W., Tarantola, S., Guillaume, J. H., Jakeman, J., Gupta, H., Melillo, N., Rabitti,
488 G., Chabridon, V., Duan, Q., Sun, X., Smith, S., Sheikholeslami, R., Hosseini, N., Asadzadeh, M., Puy,
489 A., Kucherenko, S., and Maier, H. R. (2021). The future of sensitivity analysis: An essential discipline
490 for systems modeling and policy support. *Environmental Modelling & Software*, 137:104954.

491 Reinel, T., Raúl, R., and Gustavo, I. (2019). Deep Learning Applied to Steganalysis of Digital Images: A
492 Systematic Review. *IEEE Access*, 7:68970–68990.

493 Reinel, T. S., Brayan, A. A. H., Alejandro, B. O. M., Alejandro, M. R., Daniel, A. G., Alejandro, A. G. J.,
494 Buenaventura, B. J. A., Simon, O. A., Gustavo, I., and Raúl, R. P. (2021). GBRAS-Net: A Convolutional
495 Neural Network Architecture for Spatial Image Steganalysis. *IEEE Access*, 9:14340–14350.

496 Saltelli, A., Aleksankina, K., Becker, W., Fennell, P., Ferretti, F., Holst, N., Li, S., and Wu, Q. (2019).
497 Why so many published sensitivity analyses are false: A systematic review of sensitivity analysis
498 practices. *Environmental Modelling & Software*, 114:29–39.

499 Sedighi, V., Cogranne, R., Fridrich, J., and Member, S. (2016). Content-Adaptive Steganography
500 by Minimizing Statistical Detectability. *IEEE Transactions on Information Forensics and Security*,
501 11(2):221–234.

502 Tabares Soto, R. (2016). Programación paralela sobre arquitecturas heterogéneas. *UNAL*, page 80.

503 Tabares-Soto, R., Arteaga-Arteaga, H. B., Mora-Rubio, A., Bravo-Ortíz, M. A., Arias-Garzón, D.,
504 Alzate Grisales, J. A., Burbano Jacome, A., Orozco-Arias, S., Isaza, G., and Ramos-Pollan, R. (2021).
505 Strategy to improve the accuracy of convolutional neural network architectures applied to digital image
506 steganalysis in the spatial domain. *PeerJ Computer Science*, pages 1–21.

507 Tabares-Soto, R., Ramos-Pollán, R., Isaza, G., Orozco-Arias, S., Bravo Ortíz, M. A., Arteaga Arteaga,
508 H. B., Mora Rubio, A., and Alzate Grisales, J. A. (2020). 12 - Digital media steganalysis. In Hassaballah,
509 M., editor, *Digital Media Steganography*, pages 259–293. Academic Press.

510 Tan, S., Wu, W., Shao, Z., Li, Q., Li, B., and Huang, J. (2020). CALPA-NET: Channel-Pruning-Assisted
511 Deep Residual Network for Steganalysis of Digital Images. *IEEE Transactions on Information Forensics*
512 *and Security*, 16:131–146.

513 Theodoridis, S. (2015). Chapter 18 - Neural Networks and Deep Learning. In Theodoridis, S., editor,
514 *Machine Learning*, pages 875–936. Academic Press, Oxford.

515 University, B. (2015). Steganographic algorithms. http://dde.binghamton.edu/download/stego_algorithms/.
516 2020-10-18.

517 Vasan, D., Alazab, M., Wassan, S., Safaei, B., and Zheng, Q. (2020). Image-based malware classification
518 using ensemble of cnn architectures (imcec). *Computers & Security*, 92:101748.

519 Wang, Z., Chen, M., Yang, Y., Lei, M., and Dong, Z. (2020). Joint multi-domain feature learning for
520 image steganalysis based on CNN. *Eurasip Journal on Image and Video Processing*, 2020(1).

521 Xu, G., Wu, H., and Shi, Y. (2016). Structural design of convolutional neural networks for steganalysis.
522 *IEEE Signal Processing Letters*, 23(5):708–712.

523 Ye, J., Ni, J., and Yi, Y. (2017). Deep Learning Hierarchical Representations for Image Steganalysis.
524 *IEEE Transactions on Information Forensics and Security*, 12(11):2545–2557.

525 Yedroudj, M., Comby, F., and Chaumont, M. (2018). Yedroudj-net: An efficient cnn for spatial steganaly-
526 sis. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*,
527 pages 2092–2096.

528 Zhang, R., Zhu, F., Liu, J., and Liu, G. (2019). Depth-wise separable convolutions and multi-level pooling

529   for an efficient spatial CNN-based steganalysis. *IEEE Transactions on Information Forensics and*
530   *Security*, PP(September):1–1.