electronics

MDPI

Article

Detecting Browser Drive-By Exploits in Images Using Deep Learning

Patricia Iglesias *, Miguel-Angel Sicilia * and Elena García-Barriocanal *

Computer Science Department, University of Alcala, 28805 Madrid, Spain
* Correspondence: mariapatricia.iglesi@uah.es (P.I.); msicilia@uah.es (M.-A.S.); elena.garciab@uah.es (E.G.-B.)

Abstract: Steganography is the set of techniques aiming to hide information in messages as images. Recently, stenographic techniques have been combined with polyglot attacks to deliver exploits in Web browsers. Machine learning approaches have been proposed in previous works as a solution for detecting stenography in images, but the specifics of hiding exploit code have not been systematically addressed to date. This paper proposes the use of deep learning methods for such detection, accounting for the specifics of the situation in which the images and the malicious content are delivered using Spatial and Frequency Domain Steganography algorithms. The methods were evaluated by using benchmark image databases with collections of JavaScript exploits, for different density levels and steganographic techniques in images. A convolutional neural network was built to classify the infected images with a validation accuracy around 98.61% and a validation AUC score of 99.75%.

Keywords: steganography; steganalysis; polyglots; neural networks; deep learning

Citation: Iglesias, P.; Sicilia, M.-A.; García-Barriocanal, E. Detecting Browser Drive-By Exploits in Images Using Deep Learning. *Electronics* **2023**, *12*, 473. https://doi.org/10.3390/electronics12030473

Academic Editor: Suleiman Yerima

Received: 17 October 2022
Revised: 19 December 2022
Accepted: 20 December 2022
Published: 17 January 2023

1. Introduction

Steganography is a set of techniques designed to hide information or objects by embedding them in another object called a host, so that they go unnoticed. Stenography has been applied since ancient Greece but it has recently grown in importance as an area of study in communications and computer security [1] due to its application by illegal or malicious organisations to evade security measures and extract information by hiding malicious code in digital objects such as video, images, audio or documents, among other uses. The fight against stenography is the discipline of steganalysis, which aims to detect the existence of hidden information in the host.

Depending on the steganography technique, the detection can be easier or more difficult. One of the most common techniques is the addition of a signature in the stego file, but other techniques such as adding the data after the host EOF are also very common. More sophisticated techniques distribute stego in different ways exploiting different characteristics of the images. This part is reviewed in Section 2.

Many of the current widely used tools are capable of identifying EOF steganography or signature [2], but the implementations are not as effective in detecting more sophisticated techniques and they are able to avoid the security controls.

Steganography techniques can be applied to introduce malicious code based on polyglot techniques embedded in a stego image. A polyglot is an image and JavaScript code at the same time. If in the web page it is invoked as the next block of code, an image is displayed:

```
<img src=‘‘polyglot_stego_image.jpg’’/>
```

However, if in the web page it is invoked as the next block of code, a JavaScript code embedded in the image is executed:

```
<script src=‘‘polyglot_stego_image.jpg’’> </script>
```

During the recent COVID-19 pandemic, one of the main attack categories reported by the European Union Agency for Cybersecurity (ENISA) was the delivery of malware using undetected and sophisticated mechanisms [3]. One such sophisticated type of attack is the binomial steganography-polyglot, which has been exploited for real attacks, as it is currently undetectable by standard security measures [4]. The malicious code can be executed using "polyglot" techniques, which consist of embedding the code in such a way that it is executable when is read by the web browser. In this type of attack, it is important to detect the stego image before it is executed by the browser, which requires some kind of detection model.

The attacker shows other examples of applications (https://www.bleepingcomputer.com/tag/steganography/ (accessed on 1 November 2022)), for instance, Zeus malware to set up a man-in-the-middle attack (https://www.silicon.co.uk/security/virus/zeus-banking-trojan-205640 (accessed on 1 November 2022)), Lockibot malware family to download the malicious malware as second step embedded in an image or in September of 2022, the latest Window logo cyber espionage attack (https://www.cybertalk.org/2022/09/30/hackers-hide-malware-in-windows-logo/ (accessed on 1 November 2022)) for Middle East countries.

This paper proposes a new approach to detect polyglots. It is based on the early detection of the stego image created with LSB (Least Significant Bit) steganography, LSB with Fermat and Fibonacci generators and F5 using deep learning (DL) techniques. Specifically, a convolutional neural network (CNN) is used to classify the infected images and the clean images. The main advantages of the work reported here over other approaches are (1) that the images are only resized in the pre-processing part, trying to keep the images as close as possible to the original and reducing the time processing and resources consumption; (2) a very good performance of the algorithm has been obtained in the detection of different LSB steganographies (LSB, LSB set with generator function (Fibonacci, Fermat), LSB in the description) and F5 [5]; (3) a higher quality of steganography images is able to be detected with lower relation of bits per pixel (BPP).

The rest of the paper is structured as follows: Section 2 briefly reviews the background of steganography and steganalysis techniques, including the application in polyglot attacks based on steganography. Section 3 proposes a method to detect polyglots and a description of the different setups and experiments that were performed in order to design the algorithm. Section 4 exposes the results and analysis of the different experiments. Finally, the Section 5 provides the conclusion, including remarks and outlook.

2. Background

In this section, a brief report on the the state of the art is presented, including relevant steganography techniques and the corresponding detection approaches, with a focus on the use of least significant bit stenography (LSB) and F5 for embedding polyglots. Finally, previous works applying a deep learning approach to steganalysis are described.

2.1. Stenography

As previously stated, steganography is the art of hiding information in a host in such a way that it is not detectable [6]. There are different types of steganography depending on the object where the message is hidden, e.g., text, image, audio, video and network or protocol stenography. This paper focuses on the image stenography.

Several groups of algorithms can be applied to embed data into images [7]. Some examples of groups of techniques are:

- Based on spatial domain. They are based on the statistics of the image and create a hidden channel using a replacement method. It can be implemented in a sequential way, e.g., using the least significant bits (LSB) or in a random sequence, for instance, by using the least significant bits (LSB) with Fermat or Fibonacci formulas generator (https://stegano.readthedocs.io/en/latest/software.html#the-command-stegano-lsb-set (accessed on 15 July 2021)).

- Based on the frequency domain. It spreads the data over the frequency domain of the signal. Almost all robust methods of steganography are based in the Frequency Domain. Some examples are F5 algorithm (a Discrete Cosine Transform (DCT)), OutGuess (https://www.rbcafe.es/software/outguess/ (accessed on 1 August 2022)), YASS (https://github.com/logasja/yass-js (accessed on 1 August 2022)), etc. There are more robust methods than LSB, although they have the limitation of the number of least significant bits of an image.
- Based on spread spectrum image steganography (SSIS). They are based on modulating a narrow band above the carrier.
- Based on machine learning algorithms [8].
- Manually inserting the code in the image randomly, etc.

2.2. Steganalysis: Frameworks and Techniques

Steganalysis is the process of detecting steganography by observing variations at different levels between the cover object and the final stego file. The aim of steganalysis is to identify suspicious information flows and to determine whether or not they have encoded hidden messages [9,10].

The steganalysis techniques depend on what information is available, whether it is just the stego, both the stego and the cover file, the stego and the message, or the stego and the steganography technique used. The less information available, the more difficult the steganalysis becomes. There are frameworks, such as the one proposed by Xiang-Yang et al. [11], for blind steganalysis.

In addition to the general framework, many authors suggest different taxonomies of steganalysis techniques as Nissar and Mir's one [12] or the Karampidis et al. taxonomy [13], both of them well-known and commonly accepted. Yet neither Karampidis et al. nor Nissar and Mir include the latest techniques as machine learning or deep learning as a technique to approach steganalysis, and there are only a few previous studies using these approaches to address the specificity of polyglots, which are described below.

2.3. Polyglot Attacks with Steganography

Steganography attacks are based in the broad use of multimedia files and the difficulties of the traditional security tools to detect stegos in the files. Some security approaches detect the strange behaviour, not the steganography infection, so a system could be infected for a long time without notice. The infection begins just when the user downloads the file where a polyglot is hidden. Typically, these polyglots are sent by email by phishing [4].

Polyglots are able to execute the code, e.g., in a power shell. The usual behaviour of the attack is based on botnets that works as command and control (C&C). This means that there is hidden some malicious code (downloader) which is "zombie" until the control botnet contacts to it and sends the instructions.

According to Kaspersky ICS CERT, steganography is mainly used in industry target attacks and in different areas [4]. Some of the most famous pieces of malware using steganography for espionage were Loki Bot (https://www.zdnet.com/article/lokibot-information-stealer-now-hides-malware-in-image-files/ (accessed on 1 October 2022)) and ZeroT (https://attack.mitre.org/software/S0230/ (accessed on 1 October 2022)). Other examples of bank trojans that use steganography in their attacks, some of them from the Bebloh family [14], e.g., Shiotob or URLZone, or the Ursnif family [15], e.g., Gozi or ISFB. Additionally, during the COVID-19 pandemic, there were directed attacks with polyglots related to making information about the vaccines be unnoticed.

2.4. Previous Work in Steganalysis and Deep Learning

In steganalysis, approaches outside the deep learning fields have been based nowadays in the computation of rich models followed by ensemble classifiers, as [16] or [17] propose.

Regarding deep learning, the first references using neural networks date back to 2005 [18,19]. These works proposed a feed-forward network used as classifier to detect if

there is presence of steganography or not. However, the first approaches in CNN, based in Local Contrast Normalization or Local Response Normalization, appear in 2015 [20]. The performance of this CNN for stegoanalysis was not as good as that of traditional approaches, but also in 2015, the first batch normalisation-based CNN approach emerged with similar performance to the other existing ones [21].

Since CNN requires large amounts of memory and time for the training, the steganography and AI communities worked on approaches to reduce them, resulting in approaches based in transfer learning [22], as the one proposed by [21]. However, these early approaches do not have good accuracy.

From 2015 to 2016, efforts focused on spatial steganalysis [23,24], and in 2017, work was reoriented to JPEG steganalysis [23,25]. Then came the GAN model [26,27], but it does not seem to be very successful since the accuracy was low in comparison to the traditional works. The GAN approach consisted of generating JPEG-infected images to train a CNN and generate a model that could subsequently classify between the presence of steganography or lack thereof. As it will be described in next sections, steganography is very sensitive to any modification in the images, and, as the GAN approach added additional noise in the images, the results were not as good as expected.

SRNET [28] is also a CNN for classification of images that tries to add information to help to detect singularities due to the structure of the network and get rid of the Relu. The use of Relu and a softmax function for classification, as in the model proposed in this paper, provides better results for detecting stego images, as we review in Section 4 Results and discussion.

In 2021, another CNN approach was published [29]. It uses a pre-processing stage, feature extraction, separable convolutions and a classification module. In addition to pre-processing, this approach implements a HPF (High-Pass Filter) and 30 filters consisting of padding and strides. The authors of [30] performed a study of a filter subset selection method for steganalysis CNN. This study shows that the application of redundant filters produces over-fitting, introduces noise and exploits the performance of the steganalysis CNN models. These assertions have been taken as the basis for the approach proposed in this paper, so the work reported here goes further by minimising pre-processing, only using a single rescaling step.

The metric Bits per Pixels measures the quality of the steganography implementing the relation between the number of secret bits embedded and the number of bits of the original image. This metric is used for [29] to establish the performance of the steganalysis method, being able to detect infected images with a 0.2 bpp and getting an accuracy of 80.3%. In the final model proposed in this paper, the model is able to detect images with a 0.0027 bpp and a validation accuracy of 98.61% is obtained, so it is able to detect the infection in higher-quality steganography images (with a smaller proportion of infected data in the host) and better validation accuracy.

This new approach only features resizing in the pre-processing stage, it uses a sigmoid function in the classification, instead of a softmax as [29] did, and it uses a drop-out for adding more flexibility to the model. The approach also uses Adam optimization and drop-out in order to improve the generalization of the model, as suggested in [31].

3. Proposal for a Steganalysis Approach to Polyglot Detection

3.1. Description of the Approach

The steganalysis approach reported in this paper is based on the use of convolutional neural networks (CNN) for image classification. CNNs are selected due to the performance of this type of Neural Networks (NN) in the classification of images [32,33] and the ability to learn the different dimensions of the image to distinguish the nuances of the infected images in comparison to the clean images as used for face recognition in [34].

As CNNs take images (both clean and stego) as input data for training, they are able to learn their features and classify a new image as infected or clean. It learns patterns more

difficult to identify by the visual analysis, the analysis of channels or statistical analysis. This is an important feature to take into account in the selection of this technique.

The steps that the authors have performed for the final proposal CNN are described below and shown in Figure 1. A data set has been created by collecting clean images from Coco dataset and ILSVR dataset. On the other hand, stego images were collected from StegoAppDB [35]. As only a stego F5 dataset was found, a stego dataset for polyglots must be created. Identified polyglots in Javascript were then collected. With the polyglots and the cleaned images, stegos were generated with different LSB techniques, sequential and random.

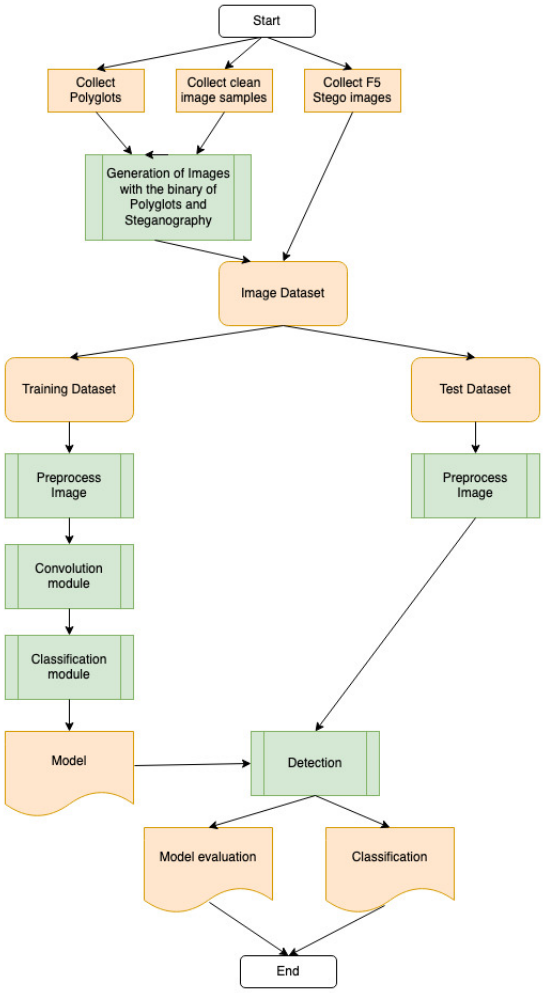

Figure 1. Framework of Blind Steganalysis.

Once that LSB stego dataset, the clean images dataset and the StegoDB App dataset have been created, the clean images and the stegos images are joined and then the images with the metadata of stego or clean label are split into two datasets: a training dataset and a test dataset. The training dataset is used to train the 2D CNN classification model and the test dataset to validate the performance of the produced model. As a result of the process, a model evaluation was obtained in which the validation accuracy was calculated, obtaining the precision in the classification of the stego images.

The model uses a rescaling pre-processing stage very light in comparison to previous tests carried out by other authors and described below. The CNN contains 10 blocks of CNN + Batch Normalisation + Relu that classify into the classes mentioned above, i.e., clean and infected. To obtain good results, a large variety of colour images, objects, different

embedded malware/polyglots and different steganography techniques were required, as explained in the following subsections.

The first model designed is based on a CNN having three parts: the preparation module, the convolution module and the classification module. The preparation module was composed of two further sub-modules in where data are re-scaled and data augmentation (rotating) and batch normalization (batch size = 32) are applied. The convolution module was composed of 10 sub-modules (conv2D, batch normalization and activation) and, finally, the classification module used a sigmoid activation on the basis that only two possible values should be handled: the presence of steganography or lack thereof.

The output of the network is the probabilities that an image belongs to the "stego" class and to the "non-stego" class. Due to poor results in validation accuracy (see Figure 2) using this model, a second version of the model was tested, dropping out the pre-processing sub-module and removing the data augmentation based on rotation. The dropping out removes part of information for the training in each iteration making more flexible the model for possible modifications and the elimination of the data augmentation reduces the errors in the training, as there were no real stego images. The second and final model (see Figure 3) obtains very good results for classification, as described in Section 4.

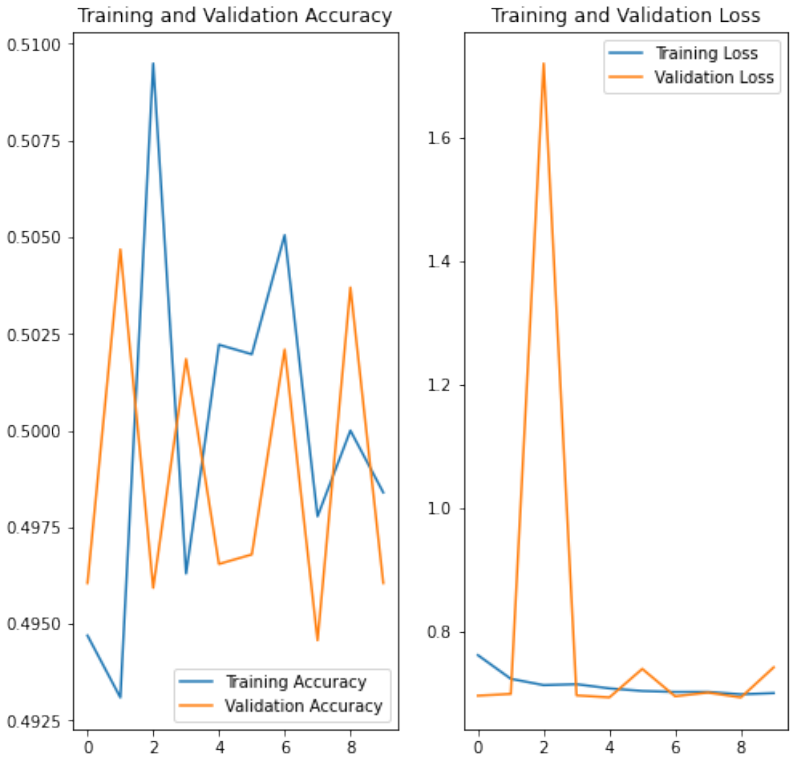

Figure 2. Validation Accuracy and Training Loss of Watermelon & Model 1 & 20 polyglots.

Regarding the dataset to train the model, watermelon images were first used in the classification model. The dataset contained both stego watermelon images generated with a polyglot using the LSB technique and clean watermelon images. Initial results gave an accuracy of 0.9672, which may suggest over-fitting. Although they will be discussed in detail in the following subsection "Experimental setup", a number of issues were identified that could potentially lead to over-fitting, namely:

- The type and variety of objects displayed in the images. COCO were used as sources since the images they contain show different types of objects. The ILSVRC dataset was also used as source of images to increase variety and avoid possible biases.
- The number of polyglots embedded in the images. Several trainings were held varying the number of Javascript polyglots and the number of images infected.

- The characteristics of the images. Training was conducted with greyscale or colour images. Polyglots were embedded before and after colour transformation for different tests.
- The homogeneity of the images (same size, orientation, etc.) Image transformation regarding size and orientation has been performed.

Multiple combinations of these cases were made to obtain higher prediction accuracy and to avoid over-fitting.

Figure 3. Structure of our proposal of CNN to detect LSB Steganography.

Data enrichment (conversion to gray, rotating and resizing the images) was also applied after the generation of the infected images, but as additional noise was introduced in the identification of the stego images, detection became impossible (accuracy of 0.561) (see Figure 4) and this approach was discarded. After testing all possible combinations in the dataset that could have an impact on accuracy and overfitting, a model with an accuracy of 95.21% (see Figure 5) was obtained. Then, the model was improved with other possible steganography algorithms (random LSB Algorithms (LSB with Fibonacci and Fermat generators and F5), and, after the training, a validation accuracy of 98.61% was obtained (see Figure 6).

The experimental setup is described in Section 3.2.

Figure 4. Validation accuracy and Training Loss of Coco Gray & Model 1 & 20 polyglots.

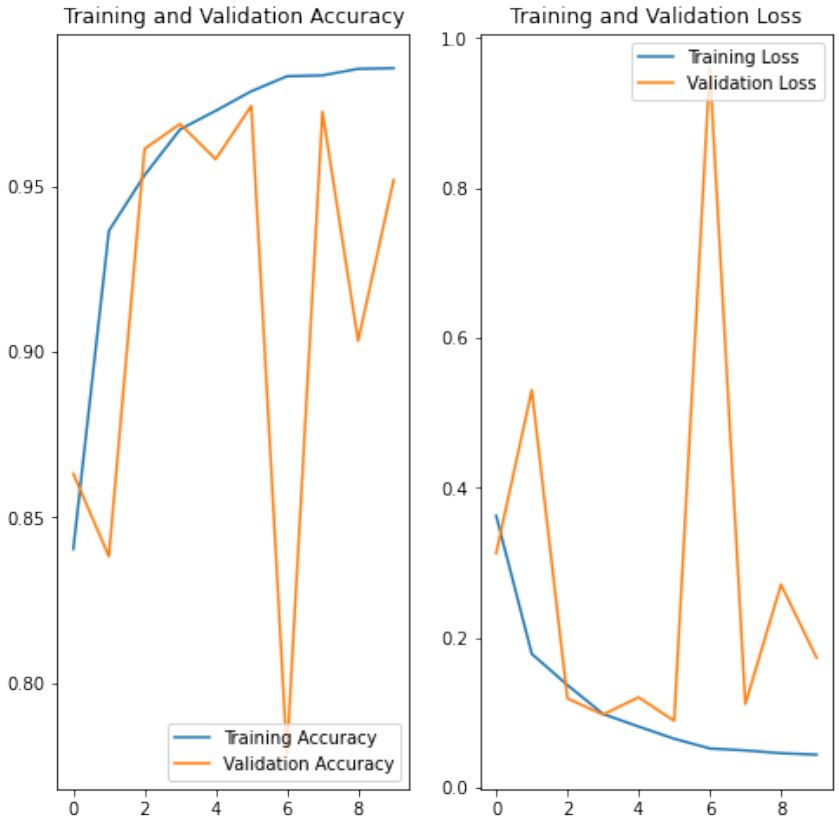

Figure 5. Validation accuracy and Training loss in Coco+ILSVR(205K) Model 2 & 104 Polyglots.

Figure 6. Validation accuracy and Validation AUC ROC in Coco+ILSVR(205K)+F5 & Model 2 & LSB, LSB Random, F5.

3.2. Experimental Setup

In order to obtain the most accurate and least over-fitted model, different datasets were created to train the model. Datasets included both clean images and stego images with known polyglots embedded using LSB stenography. Images were extracted from the COCO dataset [36] or a combination of COCO dataset and ILSVR dataset [37]. The infected datasets were composed of clean images with a different number of hidden polyglot Javascript code of known vulnerabilities [38]. More concretely, the following datasets and procedures for training the model were tested:

1. Watermelon dataset + LSB steganography (v0.1): Dataset of different watermelon images. It contained 1354 clean images and 1946 infected images with 1 polyglot.
2. Watermelon dataset + LSB steganography (v0.2): Dataset of different watermelon images. It contained 1354 clean images and 1946 infected images with 20 polyglots.
3. COCO Dataset + LSB steganography (v1): Using COCO as source from images that contain a variety of items/situations, LSB technique was used to create stego images with polyglots, resulting in a dataset with 37,000 clean images and 3000 infected images.
4. COCO Dataset + LSB steganography + Image modifications (Resizing, Relocation, ...) (v2): Using the the dataset configured in (2), data augmentation and images resizing were performed.
5. COCO Dataset + Gray Conversion + LSB Steganography (v3): Using the clean COCO dataset configured in (2) (40,000 images), 20,280 images images were first converted to greyscale and polyglots were included in 1256 of these greyscale images using the LSB technique.
6. COCO Dataset + LSB Steganography (v4): Using the images from COCO, the number of different polyglots embedded using the LSB technique was increased up to 20 common and known structures in Javascript. The number of infected images

were also increased up to 411,000 images, being 328,000 clean images and 83,000 infected images.

7. COCO Dataset + LSB Steganography (v5): As the previous dataset configuration can suggest overfitting, a new version of the training dataset was designed. Using images from COCO dataset, the number of different polyglots embedded using the LSB technique was increased up to 104 common and known structures in JavaScript. The number of clean images was reduced to 123,460 and the number of infected images to 31,000.
8. COCO Dataset + ILSVR dataset + LSB Steganography (v6): Using images from both COCO dataset and ILSVR dataset [37], the following two datasets were generated, which contained 41.026 clean images and 8.313 embedded images in the first case and 205.130 clean images and 41.026 embedded images in the second. In both cases, stego images were embedded with 104 common structures of polyglots in JavaScript using the LSB technique.
9. COCO Dataset + ILSVR dataset + LSB Steganography + LSB Steganography using Fermat and Fibonacci generation (v7): Based on v4 dataset, 33.347 images infected using Fermat and Fibonacci generator are added. The final dataset is composed of 279.503 images, from them 41.026 LSB infected images and 33.347 images infected using Fibonacci and Fermat generators.
10. COCO Dataset + ILSVR dataset + LSB Steganography + LSB Steganography using Fermat and Fibonacci generation (v7) + F5 [35] (v8) : Based on v5 dataset, 33.347 images infected using Fermat and Fibonacci generator and 621 F5 images are added. The final dataset is composed of 280.124 images, from them 41.026 LSB infected images, 621 F5 infected images and 33.347 images infected using Fibonacci and Fermat generators.

The Python library *Stegano* (https://pypi.org/project/stegano/ (accessed on 1 July 2021)) was used for generation of stegoimages. Images colour variation was implemented using the *Pillow* (https://pypi.org/project/Pillow/ (accessed on 1 July 2021)) library of Python.

Tensorflow and Python were used to generate code of the models, which were based on a convolutional neural network for the classification of images. Implementations were run in multiple environments:

- Local machine;
- Docker Virtual machine based in Tensorflow without GPU;
- Google Colab with no hardware optimizations;
- Google Colab with GPU [39];
- Google Colab with TPU.

Training in the first three environments was discarded due to the resulting long run times or the impossibility to perform the task. The number of images and the use of the neural network required hardware optimisations, and the best-performance models were obtained using Google Colab with TPU.

Regarding the hyperparametry of the models, during the different training processes of both of them, the hyperparameters were never modified in order to be able to compare the results in a robust way.

4. Results and Discussion

The two models generated (with and without pre-processing submodule) were tested. As explained above, the second model (without pre-processing submodule) was generated on the assumption that, although it was not a standard approach, getting rid of image details may condition the classification results.

The results of the validation of both models are in Table 1.

The results confirm that the pre-processing sub-module (model 1) is not needed. When the number of different polyglots that infect images is increased, the model that includes pre-processing seems not to be good (accuracy 56.1% see Figure 4).

Using the second model (without pre-processing) with a wide variety of images and the same increased number of different polyglots, accuracy reaches very good values 95.43%, see Figure 7).

Table 1. Results of Experiments.

Dataset	Model	Number of Polyglots	Type of Stego	Val Accuracy
Watermelon (v0.1)	1	1	LSB	0.9672
Watermelon (v0.2)	1	20	LSB	0.561
Coco RGB (v1)	1	1	LSB	0.9507
Coco RGB (v2)	2	20	LSB	0.9543
Coco Gray (v3)	2	20	LSB	0.9399
Coco RGB (v5)	2	104	LSB	0.9739
Coco RGB (v3+v4)	2	20	LSB + Gray	0.0915
Coco+ILSVR (v6)	2	104	LSB	1
Coco+ILSVR (v7)	2	104	LSB	0.9521
Coco+ILSVR+F5 (v8)	2	NA	LSB, F5	0.9861

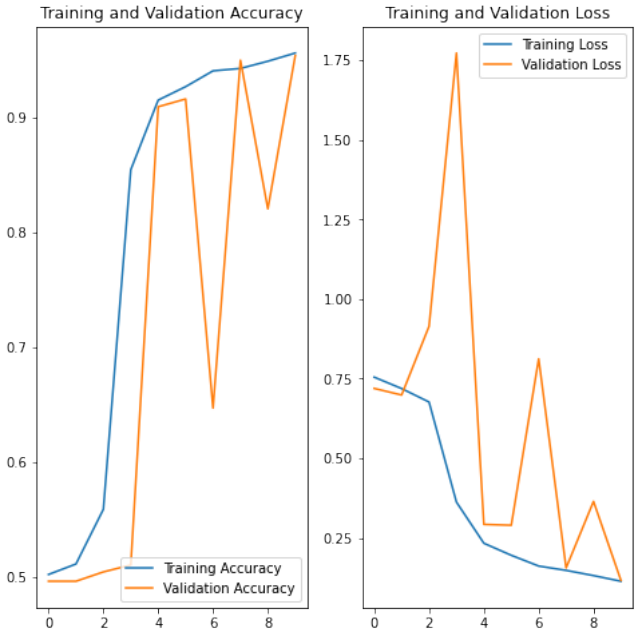

Figure 7. Validation accuracy and Training loss in Coco RGB(v2) Model 2 & 20 Polyglots.

The second model performs very well overall. The results also show that the richer the image, the easier it is to detect embedded polyglots. Thus, the second model more accurately classifies embedded polyglots in colour images than in greyscale images.

It is also noteworthy that if rescaling, greyscale conversion or rotation occurs after LSB stenography, the noise introduced makes classification not possible.

On the other hand, if a large number of different polyglots are used to infect the images, the number of images must be considerably larger to avoid overfitting.

When the second model is trained in a realistic way, i.e., with 280k (approx.) very different images that are provided from different sources and a good number of different polyglots, it can classify with 95.21% accuracy (Figure 5) whether an image is infected or not, with a 0.0027 bpp in the worse relationship, and an error less than 0.06%, which can be considered a good quality indicator of the model and it is supposed to be an advantage over other approaches such as [28,29], that obtain 80.3% of accuracy with a 0.2 bpp in the first case and a 31.3% of error in the second case.

Finally, Model 2 was trained with richer types of steganography methods (LSB + LSB

with Fermat and Fibonacci Generators and F5) and it results in a new model with a validation accuracy of 98.61% and a validation AUC score of 99.75% (see Figure 6).

5. Conclusions and Outlook

Convolutional networks have demonstrated their ability to solve image-based tasks such as recognition, classification or segmentation in previous work. In this work, these networks have been used for a stegoanalysis task, namely, for the detection of polyglot payloads in images, which is quite different from the original applications of this type of networks. Results provide evidence of the feasibility of these networks to solve the task and the model provided evidence of better detection results than other previously proposed approaches.

However, results shown here are limited to the detection of stego images using the LSB and F5 stegographic techniques. Future work should expand the range of infected images, including a diversity of stegographic techniques. This is critical in the adversarial environment of malware detection using polyglots, whereby the robustness of models for detecting a diversity of potential variations in the embedded payload represents a significant challenge.

Author Contributions: Conceptualization, M.-A.S., P.I. and E.G.-B.; methodology, M.-A.S., P.I. and E.G.-B.; software, P.I.; validation, M.-A.S., P.I. and E.G.-B.; formal analysis, M.-A.S., P.I. and E.G.-B.; investigation, M.-A.S., P.I. and E.G.-B.; resources, M.-A.S., P.I. and E.G.-B.; data curation, P.I.; writing—original draft preparation, M.-A.S., P.I. and E.G.-B.; writing—review and editing, M.-A.S., P.I. and E.G.-B.; visualization, P.I.; supervision, M.-A.S. and E.G.-B.; project administration, M.-A.S., P.I. and E.G.-B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. Kadhima, I.J.; Premaratnea, P.; Viala, P.J.; Hallorana, B. Comprehensive survey of image steganography: Techniques, Evaluations, and trends in future research. *Neurocomputing* **2019**, *335*, 299–326. [CrossRef]
2. Muñoz, A. A Simple Steganalysis Tool. Available online: https://stegsecret.sourceforge.net/ (accessed on 1 October 2022).
3. ENISA Threat Landscape 2020: Cyber Attacks Becoming More Sophisticated, Targeted, Widespread and Undetected. Available online: https://www.enisa.europa.eu/news/enisa-news/enisa-threat-landscape-2020 (accessed on 29 November 2021).
4. Steganography in Attacks on Industrial Enterprises. Available online: https://ics-cert.kaspersky.com/reports/2020/06/17/steganography-in-attacks-on-industrial-enterprises (accessed on 29 November 2021).
5. Jiang, C.; Pang, Y.; Xiong, S. A High Capacity Steganographic Method Based on Quantization Table Modification and F5 Algorithm. *Circuits Syst. Signal Process* **2014**, *33*, 1611–1626. [CrossRef]
6. Kour, J.; Verma, D. Steganography Techniques—A Review Paper. *Int. J. Emerg. Res. Manag. Technol.* **2015**, *3*, 132–135.
7. Malik, S.; Mitra, W. Hiding Information—A Survey. *J. Inf. Sci. Comput. Technol.* **2015**, *3*, 232–240.
8. Cho, D.X.; Thuong, D.T.H.; Dung, N.K. A Method of Detecting Storage Based Network Steganography Using Machine Learning. *Procedia Comput. Sci.* **2019**, *154*, 543–548. [CrossRef]
9. Wang, J.; Cheng, M.; Wu, P.; Chen, B. A Survey on Digital Image Steganography. *J. Inf. Hiding Priv. Prot.* **2019**, *1*, 87–93. [CrossRef]
10. Jiao, S.; Zhou, C.; Shi, Y.; Zou, W.; Li, X. Review on Optical Image Hiding and Watermarking Techniques. *Opt. Laser Technol.* **2019**, *109*, 370–380. [CrossRef]
11. Luo, X.Y.; Wang, D.S.; Wang, P.; Liu, F.L. A review on blind detection for image Stenography. *Signal Process.* **2008**, *88*, 2138–2157. [CrossRef]
12. Nissar, A.; Mir, A.H. Classification of Steganalysis Techniques: A Study. *Digit. Signal Process.* **2010**, *20*, 1758–1770. [CrossRef]
13. Karampidis, K.; Kavallieratou, E.; Papadourakis, G. A Review of Image Steganalysis Techniques for Digital Forensics. *J. Inf. Secur. Appl.* **2018**, *40*, 217–235. [CrossRef]
14. Bebloh—A Well-Known Banking Trojan with Noteworthy Innovations. Available online: https://www.gdatasoftware.com/blog/2013/12/23978-bebloh-a-well-known-banking-trojan-with-noteworthy-innovations (accessed on 1 October 2022).
15. Ursnif. Available online: https://attack.mitre.org/software/S0386/ (accessed on 1 October 2022).
16. Tabares-Soto, R.; Arteaga-Arteaga, H.; Mora-Rubio, A.; Bravo-Ortiz, M.A.; Arias-Garzón, D.; Grisales, J.A.A.; Jacome, A.B.; Orozco-Arias, S.; Isaza, G.; Pollan, R.R. Strategy to improve the accuracy of convolutional neural network architectures applied to digital image steganalysis in the spatial domain. *J. Comput. Sci.* **2021**, *7*, e45. [CrossRef] [PubMed]

17. Chaumont, M. Deep Learning in steganography and steganalysis from 2015 to 2018. In *Digital Media Steganography: Principles, Algorithms, Advances*; Hassaballah, M., Ed.; Elsevier: Amsterdam, The Netherlands, 2019.
18. Shi, Y.Q.; Xuan, G.R.; Zou, D.K. Image steganalysis based on moments of characteristics functions using wavelet characteristics functions using wavelet decomposition, prediction-error image, and neural network. In Proceedings of the IEEE International Conference on Multimedia and Expo, Amsterdam, The Netherlands, 6 July 2005; pp. 269–272.
19. Lie, W.N.; Lin, G.S. A Feature-based classification technique for blind steganalysis. *IEEE Trans. Multimed.* **2005**, *7*, 1007–1020
20. Tan, S.; Li, B. Stacked Convolutional Auto-Encoders for Steganalysis of Digital Images. In Proceedings of the Signal and Information Processing Association Annual Summit and Conference, Siem Reap, Cambodia, 9–12 December 2014.
21. Qian, Y.; Dong, J.; Wang, W.; Tan, T. Deep Learning for Steganalysis via Convolutional Neural Networks. In Proceedings of the Media Watermarking, Security, and Forensics, San Francisco, CA, USA, 8–12 February 2015; Volume 9404.
22. Jin, B.; Cruz, L.; Goncalves, N. Deep Facial Diagnosis: Deep Transfer Learning From Face Recognition to Facial Diagnosis. *IEEE Access* **2020**, *8*, 123649–123661. [CrossRef]
23. Boroum, M.; Chen, M.; Fridich, J. Deep Residual Network for Steganalysis of Digital Images. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 1181–1193. [CrossRef]
24. Li, B.; Wei, W.; Ferreira, A.; Tan, S. ReST-Net: Diverse Activation Modules and Parallel Subnets-Based CNN for Spatial Image Steganalysis. *IEEE Signal Process. Lett.* **2018**, *25*, 650–654. [CrossRef]
25. Xu, G. Deep Convolutional Neural Network to Detect J-UNIWARD. In Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security, Philadelphia, PA, USA, 20–22 June 2017; pp. 63–67.
26. Shi, H.; Dong, J.; Wang, W.; Qian, Y.; Zhang, X. SSGAN: Secure Steganography Based on Generative Adversarial Networks. In *Lecture Notes in Computer Science, Proceedings of the 18th Pacific-Rim Conference on Multimedia, Harbin, China, 28–29 September 2017*; Springer: Cham, Switzerland, 2017; pp. 534–544.
27. Tang, W.; Tan, S.; Li, B.; Huang, J. Automatic Steganographic Distorsion Learning Using Generative Adversarial Networks. *IEEE Signal Process. Lett.* **2017**, *24*, 1547–1551. [CrossRef]
28. Yasrab, R. SRNET: A shallow skip connection based convolutional neural network design for resolving singularities. *J. Comput. Sci. Technol.* **2019**, 34, 924–938. [CrossRef]
29. Reinel, T.S.; Brayan, A.A.H.; Alej, B.O.M.; Alej, M.R.; Daniel, A.G.; Alej, A.G.J.; Buenaventura, B.-J.A.; Simon, O.-A.; Gustavo, I.; Raúl, R.-P. GBRAS-Net: A convolutional neural network architecture for spatial image steganalysis. *IEEE Access* **2021**, *9*, 14340–14350. [CrossRef]
30. Wu, L.; Han, X.; Wen, C.; Li, B. A Steganalysis framework based on CNN using the filter subset selection method. *Multimed. Tools Appl.* **2020**, *79*, 19875–19892. [CrossRef]
31. Zheng, Q.; Yang, M.; Yang, J.; Zhang, Q.; Zhang, X. Improvement of Generalization Ability of Deep CNN via Implicit Regularization in Two-Stage Training Process. *IEEE Access* **2018**, *6*, 15844–15869. [CrossRef]
32. Liu, Y.; Dou, Y.; Qiao, P. Beyond top-N accuracy indicator: A comprehensive evaluation indicator of CNN models in image classification. *IET Comput. Vis.* **2020**, *14*, 407–414. [CrossRef]
33. Zhao, M.; Chang, C.H.; Xie, W.; Xie, Z.; Hu, J. Cloud shape classification system based on multi-channel cnn and improved fdm. *IEEE Access* **2020**, *8*, 44111–44124. [CrossRef]
34. Jin, B.; Cruz, L.; Goncalvez, N. Pseudo RGB-Face Recognition. *IEEE Sens. J.* **2022**, 22, 21780–21794. JSEN.2022.3197235. [CrossRef]
35. Newman, J.; Lin, L.; Chen, W.; Reinders, S.; Wang, Y.; Wu, M.; Guan, Y. StegoAppDB: A steganography apps forensics image database. *Electron. Imaging* **2019**, *2019*, 536. [CrossRef]
36. Lin, T.Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C.L.; Dollár, P. Microsoft COCO: Common Objects in Context. In *Lecture Notes in Computer Science, Proceedings of Computer Vision—ECCV 2014, Zurich, Switzerland, 6–12 September 2014*; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer: Cham, Switzerland, 2014.
37. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis. (IJCV)* **2015**, *115*, 211–252. [CrossRef]
38. A Collection of JavaScript Engine CVEs with PoCs. Available online: https://github.com/tunz/js-vuln-db (accessed on 29 November 2021).
39. Zhao, M.; Jha, A.; Liu, Q.; Millis, B.; Mahadevan-Jansen, A.; Lu, L.; Landman, B.; Tyska, M.J.; Huo, Y. Faster Mean-shift: GPU-accelerated clustering for cosine embedding-based cell segmentation and tracking. *Med. Image Anal.* **2021**, *17*, 102048. [CrossRef]