electronics

Article

A Generative Learning Steganalysis Network against the Problem of Training-Images-Shortage

Han Zhang [1,†], Zhihua Song [2,*,†], Qinghua Xing [2], Boyu Feng [1] and Xiangyang Lin [2]

[1] EM and UAV Engineering College, Air Force Engineering University, Xi'an 710051, China
[2] Air and Missile Defense College, Air Force Engineering University, Xi'an 710051, China
* Correspondence: szhele@163.com; Tel.: +159-2972-7388
† This authors contributed equally to this work.

Abstract: In recent years, several steganalysis neural networks have been proposed and achieved satisfactory performances. However, these deep learning methods all encounter the problem of Training-Images-Shortage (TIS). In most cases, it is difficult for steganalyses to obtain enough signals about steganography from a game opponent. In order to solve the problem of TIS for steganalysis, we propose a novel steganalysis method based on generative learning and deep residual convolutional neural networks. Comparative experiments show that the proposed architecture can achieve promising performance in response to spatial domain steganalysis despite a lack of training images.

Keywords: generative learning; steganalysis; steganography; image generation

check for updates

Citation: Zhang, H.; Song, Z.; Xing, Q.; Feng, B.; Lin, X. A Generative Learning Steganalysis Network against the Problem of Training-Images-Shortage. *Electronics* **2022**, *11*, 3331. https://doi.org/10.3390/electronics11203331

Academic Editor: Andrei Kelarev

Received: 28 August 2022
Accepted: 5 October 2022
Published: 16 October 2022

1. Introduction

Steganalysis and steganography are two sides of the same coin. Steganography attempts to achieve confidential communication by hiding secret information in a public carrier, such as an image on the internet. Steganalysis attempts to distinguish whether the communicating parties use steganography. They are not supposed to be known exactly for both players in a communication game, but they cannot be studied separately.

Steganography poses an increasing threat to network security because steganographic carriers are increasingly available on the internet and the payloads of steganography are more severe than ever before. With the widespread use of smartphones, the number of images transmitted on the internet continually grows. This provides a very convenient channel for the misuse of steganography, such as planning and coordinating criminal activities. The capabilities of steganography have also developed from hiding a small message within the Least Significant Bits (LSB) of an image to place a full-sized image within another image across all of the available bits [1]. We focus on steganalysis and detecting targets that include message-into-image steganography and image-into-image steganography. The former hides a secret text message in a cover image, whereas the latter hides a secret image in a cover image. Other types of steganography, such as hiding messages or images in voice or video recordings, are beyond the scope of this study.

Steganographic algorithms have developed from the earliest LSB, through UNIWARD (Universal Wavelet Relative Distortion) [2], WOW (Wavelet Obtained Weights) [3], HUGO [4], MG (Multivariate Generalized) [5], etc., to state-of-the-art technology based on deep learning. Deep-learning-based steganography can be divided into three types. The first type operates in an image-into-image way. The authors of [1] tried hiding a full-sized secret image within a cover image of the same size using a neural network directly in an end-to-end way. Hu et al. [6] proposed an image steganography without embedding method. The secret image was mapped into a noise vector and stego images were generated based on the noise vector by the trained generator neural network model. No modification or embedding operations were required during the process of image generation, and the information contained in the image was extracted successfully by another neural network, called the extractor, after training. The second type operates in a message-into-generated-cover way. In [7], a secure steganography method based on neural network generated images from noise and then hid the secret message in them via a traditional embedding algorithm. The third type uses the neural network as an assessment method. In [8], the neural network automatically learned embedding change probabilities for every pixel in a given spatial cover image. The learned embedding change probabilities were then converted to embedding distortions, which were adopted in the existing framework of minimal-distortion embedding. Steganographic neural networks based on adversarial training have evolved continuously and have made stego images increasingly difficult to detect, which poses a great challenge to steganalysis.

Steganalysis algorithms have also evolved from manually defined features analysis between cover images and stego images, including quality metrics [9], binary similarity [10], ensemble classifiers [11], rich model [12], embedding probabilities [13], and so on, to deep learning. Various means are used to enhance the detection accuracy of deep learning for steganography. These means include transfer learning [14,15], residual architecture [16,17], absolute values [18], channel selection [19], diverse activation [20], attention augmentation [21], feature-guided adaptation [22], etc. Deep learning is a promising framework providing state-of-the-art performance for steganalysis; however, it is generally difficult to obtain all the signals about steganographic algorithms from an adversary. Thus, TIS is an unavoidable scenario and should be dealt with in advance.

There are two ways to solve the problem of TIS, one is data augmentation [23,24] and the other is to generate new images. The former increases the training set size by transforming the available images. Typical ways include padding, rotating, re-scaling, flipping, translation, cropping, zooming, channel shuffle, dropout, and so on. However, only those that do not remove or suppress the fragile steganography signals are desirable for steganalysis applications. Indeed, these desirable ways for steganalysis include rotations by integer multiples of 90 degrees and vertical or horizontal flipping. Advanced methods include BitMix [24] and Pixels-off [25]. They focus on completely mining existing images rather than generating brand new images with different features. In this paper, we seek a breakthrough in the latter method.

The motivation behind this research was to generate new training images for spatial domain steganalysis. For convenience, we call the proposed network GLSNet, which is an abbreviation of Generative Learning Steganalysis Network. The main contributions are as follows: (1) We designed a new type of normalization layer to preserve the fragile pseudo stego signals for the generator and to accelerate the training. (2) Three shortcut connections from the input to three types of output layers were added to enhance recognition of the difference between cover image and stego image. (3) The ReLU activation function was replaced with Tanh to produce negative signals in the stego image. Such a structure is good for preserving the noise-like stego signals.

The rest of the paper is arranged as follows: In Section 2, we propose the architecture of GLSNet, dataset, the steganographic method, training details, and evaluation metrics. In Section 3, we present the experimental results. Finally, Section 4 mentions the conclusion and the future of the work.

2. Materials and Methods

The architecture of GLSNet, the dataset and steganographic method, and training details and evaluation metrics are addressed in this section.

2.1. Architecture

We began with a small set of cover images $C = \{c_i, i = 1, \cdots, n\}$ and its corresponding stego images set $S = \{s_i | s_i = F(c_i), i = 1, \cdots, n\}$, which usually lead to the problem of over-fitting for steganalysis.

Our goal was to help expand the training images dataset, which could then help to improve the diversity of the training data and delay the premature convergence of the steganalysis network. We therefore sought a neural network structure, which we called a generator $G : X \to Y$, that could learn to map a cover image to its corresponding stego image.

In theory, the optimal generator G should produce an output distribution $p_Y(\hat{y}), \hat{y} = G(x), x \in X$, identical to the empirical distribution $p_S(s), s = F(c), c \in C$. However, such a generator was not easy to train because of the extremely subtle difference between the cover and the stego–the steganography will always try to minimize the difference between the cover and stego images. These issues required a residual network, which better captures the subtle difference between the input and the output.

We designed a generator based on residual convolution blocks and added three shortcuts between the blocks to strengthen the retention of subtle differences. The details of the proposed generator are described in Section 2.

The overall architecture is called GLSNet-Generative Learning Steganalysis Network. The concept "generative learning" refers to both the training strategy of steganalysis when confronted with a training data shortage and the stego images generator in front of the steganalysis module. The proposed GLSNet, as shown in Figure 1, is composed of the following two segments: the front segment whose goal is to generate stego images, which is outlined in the figure by the left segment, and the last segment responsible for detecting the stego images, which is a deep residual convolution network called SRNet [16]. The SRNet provided state-of-the-art detection accuracy for spatial domain steganalysis and minimized the use of heuristics and externally enforced elements that are universal. If we consider steganalysis as a binary classification problem, to be an original image or a stego image, the detection accuracy is the ratio of correct classifications when the input dataset is a 1:1 mixture of cover images and stego images.

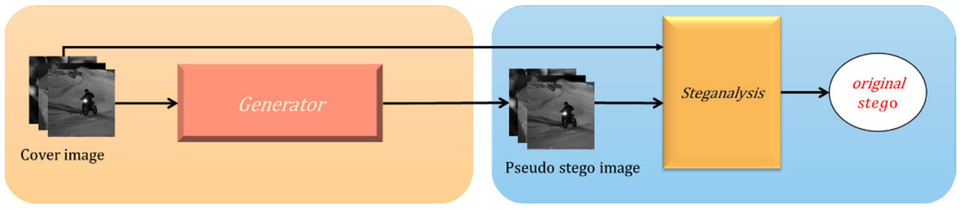

Figure 1. Architecture of the proposed GLSNet.

In order to conduct a comparative study, we tested three kinds of generators, including a fully connected generator, as shown in Figure 2, a CNN (Convolutional Neural Network) generator, as shown in Figure 3, and a residual CNN generator named RES, as shown in Figure 4, to evaluate the proposed architecture. The input of the generator was assumed to be a 256 × 256 original cover image. The output of the generator was a 256 × 256 generated stego image. The generator was trained with a small number of paired cover images and stego images. The generator was trained by reducing the error shown below (gs and c are the generated stego image and original cover image, respectively, s is the real stego image.)

$$\mathcal{L}_g(gs, s, c) = \|gs - s\|$$

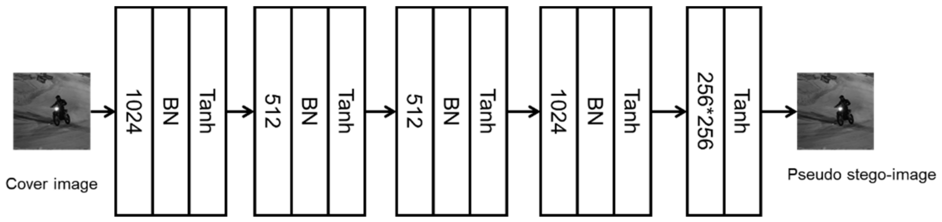

Figure 2. Structure of the fully connected generator.

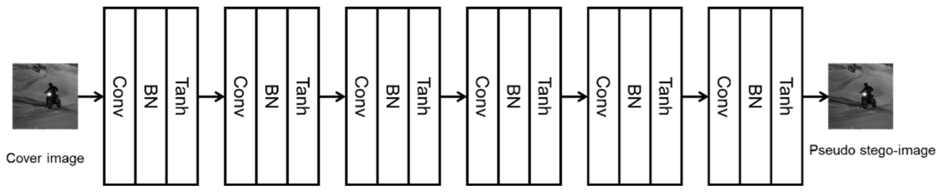

Figure 3. Structure of the CNN generator.

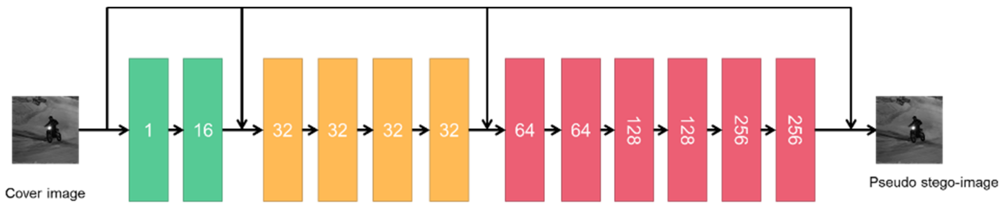

Figure 4. Structure of the RES generator.

2.2. *Details of the Generator*

The RES generator borrows heavily from SRNet [16], however, there are three improvements for this special application: (1) A new type of normalization layer was designed to preserve the fragile pseudo stego signals. (2) Three shortcut connections from the input to three types of output layers were added to enhance the learning of the difference between cover image and stego image, as depicted in Figure 4. (3) The ReLU activation function was replaced with Tanh to produce negative signals as in the stego image. Such a structure can better preserve the noise-like stego signals.

The RES generator consisted of 12 layers, as shown in Figure 4, and the details of each layer of the generator are shown in Table 1.

Table 1. Details of each layer of the generator.

Input	$1 \times 256 \times 256$
Layer1	$\text{Conv2D}(1, 16, 3, 1, 1) \rightarrow$IVN (16) $\rightarrow$ Tanh
Layer2	$\text{Conv2D}(16, 32, 3, 1, 1) \rightarrow$IVN (32) $\rightarrow$ Tanh
Layer3	$\text{Conv2D}(32, 32, 1, 1, 0) \rightarrow$IVN (32) $\rightarrow$ Tanh $\rightarrow \text{Conv2D}(16, 32, 3, 1, 1) \rightarrow$IVN (32) $\rightarrow$ Tanh
Layer4	$\text{Conv2D}(32, 32, 1, 1, 0) \rightarrow$IVN (32) $\rightarrow$ Tanh $\rightarrow \text{Conv2D}(32, 32, 1, 1, 0) \rightarrow$IVN (32) $\rightarrow$ Tanh
Layer5	$\text{Conv2D}(32, 32, 1, 1, 0) \rightarrow$IVN (32) $\rightarrow$ Tanh $\rightarrow \text{Conv2D}(32, 32, 1, 1, 0) \rightarrow$IVN (32) $\rightarrow$ Tanh
Layer6	$\text{Conv2D}(32, 32, 1, 1, 0) \rightarrow$IVN (32) $\rightarrow$ Tanh $\rightarrow \text{Conv2D}(32, 64, 3, 1, 1) \rightarrow$IVN (64) $\rightarrow$ Tanh
Layer7	$\text{Conv2D}(64, 64, 1, 1, 0) \rightarrow$IVN (64) $\rightarrow$ Tanh $\rightarrow \text{Conv2D}(64, 64, 1, 1, 0) \rightarrow$IVN (64) $\rightarrow$ Tanh $\rightarrow$ maxPool2D (3,1)
Layer8	$\text{Conv2D}(64, 64, 1, 1, 0) \rightarrow$IVN (64) $\rightarrow$ Tanh $\rightarrow \text{Conv2D}(64, 128, 1, 1, 0) \rightarrow$IVN (128) $\rightarrow$ Tanh $\rightarrow$ maxPool2D (3,1)
Layer9	$\text{Conv2D}(128, 128, 1, 1, 0) \rightarrow$IVN (128) $\rightarrow$ Tanh $\rightarrow \text{Conv2D}(128, 128, 1, 1, 0) \rightarrow$IVN (128) $\rightarrow$ Tanh $\rightarrow$ maxPool2D (3,1)
Layer10	$\text{Conv2D}(128, 128, 1, 1, 0) \rightarrow$IVN (128) $\rightarrow$ Tanh $\rightarrow \text{Conv2D}(128, 256, 3, 1, 1) \rightarrow$IVN (256) $\rightarrow$ Tanh $\rightarrow$ maxPool2D (3,1)
Layer11	$\text{Conv2D}(256, 256, 1, 1, 0) \rightarrow$IVN (256) $\rightarrow$ Tanh $\rightarrow \text{Conv2D}(256, 256, 1, 1, 0) \rightarrow$IVN (256) $\rightarrow$ Tanh $\rightarrow$ maxPool2D (3,1)
Layer12	$\text{Conv2D}(256, 256, 1, 1, 0) \rightarrow$IVN (256) $\rightarrow$ Tanh $\rightarrow \text{Conv2D}(256, 1, 1, 1, 0) \rightarrow$IVN (1) $\rightarrow$ Tanh $\rightarrow$ maxPool2D(3,1)
Output	$1 \times 256 \times 256$
Notes	• Conv2D(I,O,K,S,P) is a convolution layer, where I stands for the number of input channels, O stands for the number of output channels, K stands for the kernel size, S stands for the stride number, and P stands for the padding size. • IVN (32) stands for an instance variance normalization layer with 32 channels. • Tanh stands for the Tanh activation layer. • maxPool2D (3,1) stands for a maximize pool layer, where the kernel size is 3 and padding is 1.

Existing normalization techniques, such as Batch Normalization [26], Layer Normalization [27], Instance Normalization [28], Group Normalization [29], etc., were undesirable for the generator because they changed the mean and variance of the input simultaneously, resulting in destructive proportional distortion, which, in other words, is the destruction of the prior conditions for the generation of pseudo steganography signals.

We designed a new normalization method, IVN: instance variance normalization, and it achieved more satisfactory results in the generation of pseudo steganography images.

The input of a network layer can be expressed as a multidimensional array

$$Input = X_{ijkl}, i \le N, j \le C, k \le H, l \le W$$

where $N(N < M)$ is the number of images in a batch, C is the number of channels for each image, H is the height of the image, and W is the width of the image.

The mean of an instance $x_{ij} = \left(x^{(1,1)}, \cdots, x^{(H,W)}\right)$ is

$$\mu = \frac{1}{HW} \sum_{k=1}^{H} \sum_{l=1}^{W} x^{(k,l)}$$

and its variance is

$$\sigma^2 = Var\left[x_{ij}\right] = \frac{1}{HW} \sum_{k=1}^{H} \sum_{l=1}^{W} \left(x^{(k,l)} - \mu\right)^2$$

We will normalize each pixel of the instance

$$y^{(k.l)} = \frac{x^{(k,l)}}{\sqrt{\sigma^2 + \epsilon}}$$

where ϵ is a small value that prevents the denominator from being zero.

The IVN transform was a differentiable transformation that introduced variance normalized activations into the network. The gradient was computed using the chain rule as follows:

$$\frac{\partial Loss}{\partial \mu} = \sum_{k=1}^{H} \sum_{l=1}^{W} \frac{\partial Loss}{y^{(k.l)}} \cdot \frac{-1}{\sqrt{\sigma^2 + \epsilon}}$$

$$\frac{\partial Loss}{\partial \sigma^2} = \sum_{k=1}^{H} \sum_{l=1}^{W} \frac{\partial Loss}{y^{(k.l)}} \cdot \left(x^{(k,l)} - \mu\right) \cdot \frac{-1}{2} \left(\sigma^2 + \epsilon\right)^{-\frac{3}{2}}$$

$$\frac{\partial Loss}{\partial x^{(k,l)}} = \frac{\partial Loss}{\partial y^{(k.l)}} \cdot \frac{-1}{\sqrt{\sigma^2 + \epsilon}} + \frac{\partial Loss}{\partial \sigma^2} \cdot \frac{2\left(x^{(k,l)}\right)}{HW} + \frac{\partial Loss}{\partial \mu} \frac{1}{m}$$

Unlike the SRNet, we used Tanh as the activation function because it matched the residuals better as it may be positive or negative. The formula of Tanh is as follows:

$$tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

2.3. Dataset and Steganographic Method

The proposed architecture was evaluated and contrasted with different generators on the commonly used publicly available source BOSSBase 1.01, which contains 10,000 grayscale images selected from seven types of cameras.

The experiments were executed for S-UNIWARD and Baluja-Net [1] to cover both the classic steganographic algorithm and the state-of-the-art deep-learning-based embedding network. S-UNIWARD is a well-known content adaptive steganographic algorithm that hides a small message within the texture or noisy regions of a larger image. Baluja-Net is a deep-learning-based steganography method that can place a full-sized image within another image of the same size. Unlike many popular steganographic methods that encode the secret message within the LSB of the cover image, Baluja-Net compresses and distributes the secret image's representation across all of the available bits.

We place greater emphasis on Baluja-Net because the deep learning steganographic method is highly variable and makes it more difficult to obtain highly authentic cover and stego image pairs compared with other low payload classic algorithms. In order to prepare training stego images for the generator, Baluja-Net was trained for 100 epochs with a batch size of 2. The training details are explained below: For an image i, we randomly selected any other image $j \in \{k|1 \leq k \leq 10,000, k \neq i, k \in Z\}$ as a secret image for steganography. In this way, 5000 real stego images were generated. An Adam optimizer was used with a learning rate $lr = 0.002$ and coefficient $betas = (0.5,\ 0.999)$. Baluja contained a hiding network H and a reveal network R. After about forty iterations of training on the full dataset of the original cover images, the loss values of these two networks tended to stabilize and converge, as shown in Figure 5.

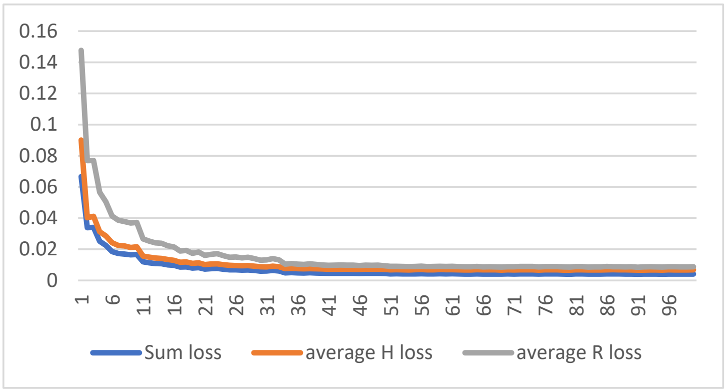

Figure 5. Loss of network H, network R, and the total loss of these two networks in the deep neural network with different iterations of training.

Figure 6 shows different scenes selected from BOSSBase. The third column is the residual between the cover and the stego, and these residuals are focused mainly on the position with a complex texture.

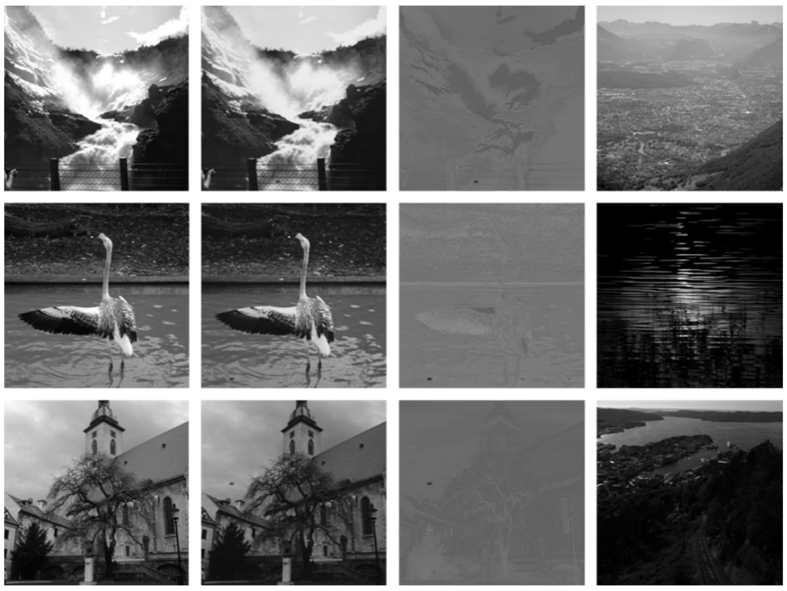

Figure 6. Samples generated by Baluja-Net from several categories in BOSSBase dataset. The first column is the original cover image. The second column is the stego images. The third column is the difference value between the stego image and cover image. The fourth column is the secret image that embedded with the cover image to make a stego image.

2.4. Parameters and Evaluation Metric

The network was trained on a PC with an Intel (R) Core(TM) i7-9700K CPU @ 3.60GHZ, 32GB DDR4 memory, graphics processing unit (GPU) NVIDIA GeForce RTX2080Ti, and 11 GB of memory.

Due to limited hardware performance, the input size of our model was set to 256 × 256. Thus, we resized images of BOSSBase from their original size 512 × 512 to 256 × 256 using cv2.resize with "interpolation = INTER_NEAREST". We generated 5000 cover and stego image pairs using S-UNIWARD and Baluja-Net each, and split them into the following two sets~: 2 × 300 cover and stego image pairs for training; and 2 × 4700 for testing.

The performance of the GLSNet was measured with the total classification error probability on the testing set,

$$P = (N_{cs} + N_{sc})/4700$$

where N_{cs} is the number of cover images that are identified as stego images, and N_{sc} is the number of stego images that are identified as cover images.

3. Results

We conducted several comparative experiments and display the results in Table 2. "Non" in the second column represents the steganalyses trained with the real cover and stego image pairs without using the pseudo stego image.

Table 2. Performance comparisons of proposed GLSNet with different generators and without generator.

Steganography	**Generator**	**No. of Original Cover-Stego Image Pairs**							
		1	**2**	**5**	**10**	**20**	**50**	**100**	**300**
Baluja-Net	RES	77.75%	81.92	86.33%	88.11%	89.08%	90.75%	85.58%	91.33%
	FC	50.00%	50.00%	50.00%	50.12%	50.26%	51.51%	50.21%	51.18%
	CNN	50.56%	51%	55.69%	63.32%	64.96%	66.26%	67.56%	68.66%
	Non	62.83%	68.17%	71.37%	79.92%	79.68%	94.78%	95.00%	92.59%
S_UNIWARD (0.2 bpp)	RES	59.37%	59.93%	63.38%	64.05%	69.36%	72.73%	81.50%	87.75%
	FC	50.00%	50.00%	50.00%	50.00%	50.11%	50.15%	51.25%	51.63%
	CNN	50.00%	50.00%	50.00%	50.00%	52.43%	58.29%	59.58%	58.66%
	Non	50.00%	50.00%	52.13%	52.59%	58.01%	59.35%	67.57%	83.39%
S_UNIWARD (0.4 bpp)	RES	63.17%	63.68%	66.83%	64.17%	71.00%	79.33%	88.50%	91%
	FC	50.00%	50.00%	50.00%	50.00%	51.17%	51.11%	52.00%	52.36%
	CNN	50.00%	50.00%	50.00%	50.05%	60.36%	65.13%	65.5%	63.33%
	Non	56.61%	58.33%	57.83%	60.54%	68.17%	74.60%	92.00%	98.51%

As shown in Table 1, the proposed RES generator performs better than the FC generator, CNN generator, and non-generative training mode. All the models perform better with Baluja-Net than with S-UNIWARD. This is because a full-sized image is embedded in a cover image with Baluja-Net, which increases its payload to up to 8 bit/pixel (bpp) whereas the payload of S-UNIWARD was below or equal to 0.4 bpp in the experiment. Compared with the non-generative training mode, the proposed generative learning architecture achieves a significant performance level when the number of cover-stego image pairs is small. This result is consistent with the theoretical viewpoints. If fewer real cover and stego image pairs are used to train the steganalysis network, the generalization ability and detection accuracy of the steganalysis network will be greatly restricted. In contrast, the generated image dataset may have abundant texture expressions, which has the potential to increase the generalization ability of the steganalysis network and its detection accuracy; our experiments have confirmed this.

In addition, we also examined the impact of quality and noise of the original cover images on the detection performance in preliminary experiments. We used filtering operations to reduce the quality of the cover, or add noise to reduce the signal-to-noise ratio (SNR) of the cover. These low quality, or low SNR, cover images were then used to generate stego images in the same way as their originals. These generated cover-stego image pairs were then used to train the proposed GLSNet in the same configuration depicted in Section 2. Experimental results show that the quality and SNR of original cover images do not affect the performance of GLSNet. This is because the generator extracts the residual information between the cover and the stego, and this information does not relate to the quality and SNR of the cover.

However, if the image quality or SNR is changed on cover or stego only, the generator will recognize the change as residual, which will have a great impact. Strictly speaking, such image pairs are incomplete, which could be the object of future research.

4. Discussion

It is difficult to obtain enough signals about deep-learning-based steganography from the point of view of game theory. Thus, the problem of the Training-Images-Shortage is prevalent for the deep learning steganalysis in deep learning steganography. We proposed a network we call GLSNet to solve the problem of TIS for the steganalysis and the results show that it performs well in solving such a problem. shown in the Table 1, even with just one pair of cover and stego images, GLSNet achieves a 77.75% detection accuracy compared with the detection accuracy of 62.83% of the nongenerative training paradigm.

The generative learning architecture helps solve the problem of TIS and improves the detection accuracy of the steganalysis when it just has a small number of training images. The concept of generative learning could also be applied to other situations where it is not easy to obtain training data in a confrontational environment, such as anomaly detection in network security, the recognition of an enemy target in the radar, etc.

Although this paper demonstrates the feasibility of generative learning for image hidden steganalysis, there is still much work to complete. Future research directions include the development of new generative learning network structures and the improvement of their capability to mimic more steganographic methods to generate training stego images.

Author Contributions: Conceptualization, H.Z. and Z.S.; methodology, H.Z.; software, Z.S.; validation, Q.X.; formal analysis, H.Z. and Z.S.; investigation, B.F.; resources, X.L.; data curation, Z.S.; writing—original draft preparation, H.Z.; writing—review and editing, Z.S.; visualization, Z.S.; supervision, Q.X.; project administration, Q.X.; funding acquisition, Q.X. and X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 71571190, 62002381.

Acknowledgments: The authors are grateful to Feng Chen, Fuxian Liu and Yongmei Zhao for their useful advice during the preparation and writing of the paper. We thank the associate editor and the reviewers for their valuable comments that improved this paper.

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

References

1. Baluja, S. Hiding Images in Plain Sight: Deep Steganography. In *Advances in Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2017; Volume 30. Available online: https://proceedings.neurips.cc/paper/2017/file/838e8afb1ca34354ac209f53d90c3a43-Paper.pdf (accessed on 8 August 2022).
2. Holub, V.; Fridrich, J.J.; Denemark, T. Universal distortion function for steganography in an arbitrary domain. *EURASIP J. Inf. Secur.* **2014**, *2014*, 1–13. [CrossRef]
3. Holub, V.; Fridrich, J. Designing steganographic distortion using directional filters. In Proceedings of the 2012 IEEE International Workshop on Information Forensics and Security (WIFS), Costa Adeje-Tenerife, Spain, 2–5 December 2012; pp. 234–239.
4. Filler, T.; Fridrich, J. Gibbs Construction in Steganography. *IEEE Trans. Inf. Forensics Secur.* **2010**, *5*, 705–720. [CrossRef]
5. Sedighi, V.; Fridrich, J.; Cogranne, R. Content-adaptive pentary steganography using the multivariate generalized Gaussian cover model. In *Media Watermarking, Security, and Forensics*; SPIE: San Francisco, CA, USA, 2015; Volume 9409, pp. 144–156. [CrossRef]
6. Hu, D.; Wang, L.; Jiang, W.; Zheng, S.; Li, B. A Novel Image Steganography Method via Deep Convolutional Generative Adversarial Networks. *IEEE Access* **2018**, *6*, 38303–38314. [CrossRef]

7. Shi, H.; Dong, J.; Wang, W.; Qian, Y.; Zhang, X. SSGAN: Secure Steganography Based on Generative Adversarial Networks. In *Advances in Multimedia Information Processing—PCM 2017*; Springer International Publishing: Cham, Switzerland, 2018; pp. 534–544.
8. Tang, W.; Tan, S.; Li, B.; Huang, J. Automatic Steganographic Distortion Learning Using a Generative Adversarial Network. *IEEE Signal Process. Lett.* **2017**, *24*, 1547–1551. [CrossRef]
9. Avcibas, I.; Memon, N.D.; Sankur, B. Steganalysis using image quality metrics. *IEEE Trans. Image Process.* **2003**, *12 2*, 221–229. [CrossRef]
10. Avcıbaş, İ.; Kharrazi, M.; Memon, N.; Sankur, B. Image Steganalysis with Binary Similarity Measures. *EURASIP J. Adv. Signal Process.* **2005**, *2005*, 679350. [CrossRef]
11. Kodovský, J.; Fridrich, J.J.; Holub, V. Ensemble Classifiers for Steganalysis of Digital Media. *IEEE Trans. Inf. Forensics Secur.* **2012**, *7*, 432–444. [CrossRef]
12. Denemark, T.; Sedighi, V.; Holub, V.; Cogranne, R.; Fridrich, J. Selection-channel-aware rich model for Steganalysis of digital images. In Proceedings of the 2014 IEEE International Workshop on Information Forensics and Security (WIFS), Atlanta, GA, USA, 3–5 December 2014; pp. 48–53. [CrossRef]
13. Tang, W.; Li, H.; Luo, W.; Huang, J. Adaptive Steganalysis against WOW Embedding Algorithm. In Proceedings of the 2nd ACM Workshop on Information Hiding and Multimedia Security, Salzburg, Austria, 11–13 June 2014; pp. 91–96. [CrossRef]
14. Dengpan, Y.; Shunzhi, J.; Shiyu, L.; ChangRui, L. Faster and transferable deep learning steganalysis on GPU. *J. Real-Time Image Process.* **2019**, *16*, 623–633. [CrossRef]
15. Padmasiri, A.T.; Hettiarachchi, S. Impact on JPEG Image Steganalysis using Transfer Learning. In Proceedings of the 2021 10th International Conference on Information and Automation for Sustainability (ICIAfS), Negombo, Sri Lanka, 3–5 May 2021; pp. 234–239. [CrossRef]
16. Boroumand, M.; Chen, M.; Fridrich, J.J. Deep Residual Network for Steganalysis of Digital Images. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 1181–1193. [CrossRef]
17. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
18. Xu, G.; Wu, H.; Shi, Y.Q. Structural Design of Convolutional Neural Networks for Steganalysis. *IEEE Signal Process. Lett.* **2016**, *23*, 708–712. [CrossRef]
19. Yang, J.; Liu, K.; Kang, X.; Wong, E.; Shi, Y. Steganalysis based on awareness of selection-channel and deep learning. In *Digital Forensics and Watermarking—16th International Workshop, IWDW 2017*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 263–272. [CrossRef]
20. 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]
21. Huang, S.; Zhang, M.; Ke, Y.; Bi, X.; Kong, Y. Image steganalysis based on attention augmented convolution. *Multimed. Tools Appl.* **2022**, *81*, 19471–19490. [CrossRef]
22. Zhang, L.; Abdullahi, S.M.; He, P.; Wang, H. Dataset mismatched steganalysis using subdomain adaptation with guiding feature. *Telecommun. Syst.* **2022**, *80*, 263–276. [CrossRef]
23. Itzhaki, T.; Yousfi, Y.; Fridrich, J. Data Augmentation for JPEG Steganalysis. In Proceedings of the 2021 IEEE International Workshop on Information Forensics and Security (WIFS), Montpellier, France, 7–10 December 2021; pp. 1–6. [CrossRef]
24. Yu, I.-J.; Ahn, W.; Nam, S.-H.; Lee, H.-K. {BitMix}: Data augmentation for image steganalysis. *Electron. Lett.* **2020**, *56*, 1311–1314. [CrossRef]
25. Yedroudj, M.; Chaumont, M.; Comby, F.; Amara, A.O.; Bas, P. Pixels-off: Data-augmentation Complementary Solution for Deep-learning Steganalysis. In Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security, Denver, CO, USA, 22–24 June 2020; pp. 39–48.
26. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; Volume 37, pp. 448–456. Available online: https://proceedings.mlr.press/v37/ioffe15.html (accessed on 8 August 2022).
27. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer Normalization. 2016. Available online: http://arxiv.org/abs/1607.06450 (accessed on 8 August 2022).
28. Ulyanov, D.; Vedaldi, A.; Lempitsky, V.S. Instance Normalization: The Missing Ingredient for Fast Stylization. *CoRR* **2016**, abs/1607.0. Available online: http://arxiv.org/abs/1607.08022 (accessed on 8 August 2022).
29. Wu, Y.; He, K. Group Normalization. In *Computer Vision—ECCV 2018*; Springer International Publishing: Cham, Switzerland, 2018; pp. 3–19.