Enhancing Image Steganalysis with Adversarially Generated Examples

Kevin Alex Zhang and Kalyan Veeramachaneni

MIT, Cambridge MA 02139, USA
{kevz,kalyanv}@mit.edu

Abstract. The goal of image steganalysis is to counter steganography algorithms which attempt to hide a secret message within an image file. We focus specifically on blind image steganalysis in the spatial domain which involves detecting the presence of secret messages in image files without knowing the exact algorithm used to embed them. In this paper, we demonstrate that we can achieve better performance on the blind steganalysis task by training the YeNet architecture with adversarially generated examples provided by SteganoGAN.

Keywords: Steganalysis · Steganography · Deep Learning

1 Introduction

Modern image steganography has found applications in everything from malware, where it can be used to transmit command-and-control instructions, to industrial espionage, where it can be used to hide or exfiltrate information. Unlike cryptography, which attempts to hide the content of the message, steganography attempts to hide the presence of the message itself by embedding it within otherwise benign content.

To combat image steganography, we can turn to image steganalysis algorithms which attempt to detect steganographic images. In general, these techniques work by analyzing the image file and identifying statistical anomalies in the pixel value distribution. Examples of steganalysis techniques include Primary Sets [4], RS analysis [7], Sample Pairs [5], and Spatial Rich Models [6]. Recently, new deep learning-based techniques have been developed for this task and have achieved state-of-the-art detection rates [11].

At the same time, new deep learning-based techniques have been developed for image steganography, yielding impressive results and achieving higher relative payloads as in [1, 10, 8, 13, 12]. Interestingly, since deep learning-based techniques are learned from data (and a random initial state), a new instance of a deep learning-based technique can be created simply by re-training the model, significantly reducing the cost of inventing a "new" steganography algorithm. This further complicates our analysis since an effective steganalysis algorithm now must not only defeat a specific instance of a steganography model but must, in fact, defeat all possible instances. In this paper, we will focus specifically on two

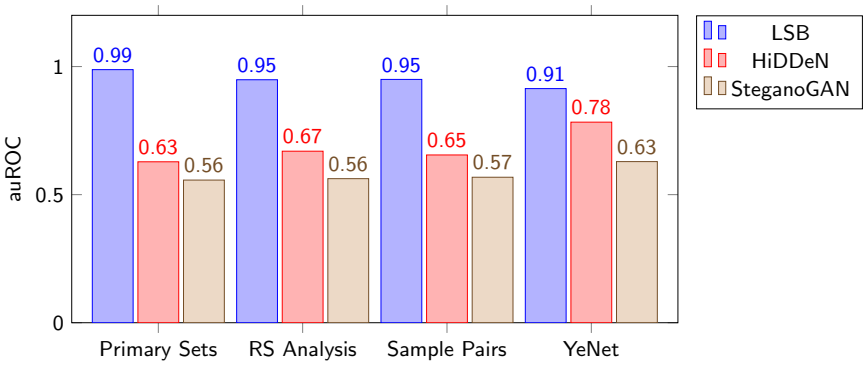

Fig. 1: This figure shows the performance of four different steganalysis techniques on steganographic images produced by the Least Significant Bits (LSB) algorithm, HiDDeN [13], and SteganoGAN [12]. We examine three different static (e.g. non-trainable) spatial steganalysis tools as well as one deep learning-based method (YeNet) that was trained to detect LSB images. Based on these results, we see that none of these techniques are particularly effective at detecting steganographic images generated by HiDDeN or SteganoGAN, suggesting that models trained on LSB steganography do not generalize well to deep learning-based steganography.

techniques, HiDDeN [13] and SteganoGAN [12], which use generative adversarial networks to create hard-to-detect steganographic images.

Due to the simultaneous development of both improved steganography methods and improved steganalysis algorithms, we start by running a simple experiment to determine the current state of steganography and steganalysis. We generated steganographic images using the least significant bits (LSB) algorithm, HiDDeN, and SteganoGAN; then, we used multiple steganalysis tools to try and detect these steganographic images. Figure 1 presents the performance of these steganalysis tools and we observe that although all the models are capable of detecting images generated using the simple least significant bits algorithm, none of them excel at detecting steganographic images generated by HiDDeN or SteganoGAN. The results from this experiment raise several questions:

1. Existing steganalysis algorithms are not effective at detecting steganographic images generated by methods not represented in the training set. Can we overcome this limitation and build steganalysis systens that generalize better?

2. We can easily create new instances of deep learning-based steganography algorithms by re-training the model with a different initial state. Can we take advantage of the fact to build more robust steganalysis systems?

To address these questions, we will borrow existing steganography and steganalysis architectures and use them to investigate the implications of being able to create new "instances" of steganography algorithms. In addition, we will

Fig. 2: Examples of cover images (top) and the corresponding steganographic images generated using LSB (middle) and SteganoGAN (bottom) with a relative payload of one bit per pixel. Both steganography techniques produce high quality images which, to the human eye, appear identical to the cover images.

examine the behavior of the YeNet architecture as we change the composition of the training dataset and evaluate its ability to generalize to previously unseen steganography algorithms.

The paper is organized as follows: Section 2 presents the different steganography methods used to generate our dataset, Section 3 presents the different steganalysis methods used in our experiments, Section 4 describes our experimental setup, and Section 5 presents our results.

2 Steganography

The standard image steganography task involves two operations: encoding and decoding. The encoding operation takes a cover image and a secret message and combines them to create a steganographic image which closely resembles the cover image but has the secret message hidden inside. Then, after the steganographic image is transmitted to the recipient, the decoding operation is applied to the steganographic image and the secret message is extracted.

In our experiments, we will be using three different techniques to generate steganographic images: least significant bits (LSB), HiDDeN [13], and SteganoGAN [12]. Examples of stenographic images generated by some of these techniques are shown in Figure 2. Both HiDDeN and SteganoGAN use convolutional neural networks and adversarial training to learn to produce realistic steganographic images.

2.1 Least Significant Bits

The simplest way to embed data in images is to simply swap the least significant bit of each pixel with the corresponding data bit. For example, given a standard

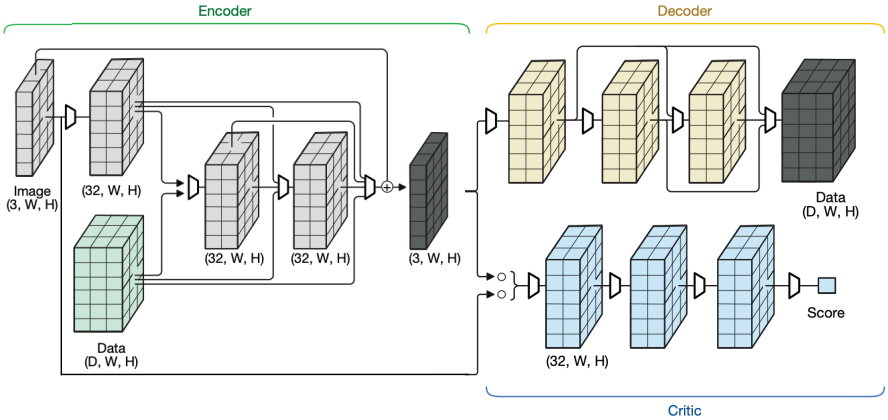

Fig. 3: The SteganoGAN architecture, reproduced with modifications from [12]. The encoder module maps a data tensor and a cover image to a steganographic image, the decoder module maps a steganographic image to a data tensor, and the critic module provides feedback on the quality of the steganographic image. The trapezoids represent convolutional blocks which consist of a convolutional layer, a leaky ReLU activation function, and a batch normalization operation. Two or more arrows merging represent concatenation operations and the curly bracket represents a batching operation.

RGB image and a sequence of data bits, we can simply loop from left-to-right, from top-to-bottom, and over the color channels and replace the lowest bit of each pixel value with the data bit. This naive approach is well-known by practitioners, easy to create paired datasets for, and simple to defend against. Examples of steganographic images generated by this technique are presented in the middle row of Figure 2. We will use steganographic images generated by this technique to initialize our steganalysis models and provide a baseline.

2.2 HiDDeN

The first deep learning-based steganography algorithm we will examine is HiDDeN [13]. This model is designed to take a fixed-length bit vector and an arbitrarily sized cover image and produce a steganographic image; note that a given model is only capable of embedding a fixed number of bits into an image, regardless of the size of the image. The HiDDeN architecture uses convolutional neural networks to represent (1) the encoder, which learns to take the image and bit vector and produces a steganographic image, (2) the decoder, which learns to take the steganographic image and decode the bit vector, and (3) the adversary, which learns to detect steganographic images and provides feedback to the encoder on how to avoid detection.

2.3 SteganoGAN

We will also use SteganoGAN, a competing deep learning-based steganography technique proposed in [12], to generate steganographic images. Examples of steganographic images generated by this technique are presented in the bottom row of Figure 2. This model is conceptually similar to [13] and features a similar encoder-decoder-critic architecture, but is able to scale more effectively to larger images while maintaining a constant embedding rate. The SteganoGAN architecture, shown in Figure 3, is designed to take in a data tensor and an arbitrarily sized cover image and create a steganographic image. Unlike the HiDDeN architecture where the data vector is of a fixed length, the the size of the data tensor in SteganoGAN scales with the size of the cover image so that larger cover images will naturally be able to hold more data.

3 Steganalysis

Compared to the steganography task, the steganalysis task seems simple: given an image, the goal is to identify whether it is cover image or a steganographic image. We use two steganalysis tools, StegExpose [3] and YeNet [11], to evaluate our ability to detect steganographic images. The former is a collection of static steganalysis tools which does not need to be trained, whereas the latter is intended to be trained on datasets containing examples of cover and steganographic images.

3.1 StegExpose

To set a baseline for detecting steganographic images, we use StegExpose [3], a popular steganalysis tool which implements several different steganalysis algorithms including Primary Sets [4], RS analysis [7], and Sample Pairs [5]. We generate steganographic images using each of the techniques discussed in the previous section and report the detection performance as measured by the area under the receiver operating characteristic in Figure 1. Based on these results, we see that the steganalysis algorithms all excel at detecting steganographic images generated using the least significant bit algorithm but fail to detect images generated by either HiDDeN or SteganoGAN.

3.2 YeNet

We also experiment with the YeNet architecture from [11], which can be trained on paired datasets containing examples of cover images and the corresponding steganographic images generated by various models. The YeNet architecture is similar to standard image classification architectures but the first set of convolutional layers are manually set to extract Spatial Rich Model [6] features. Spatial Rich Models are, in their own right, an effective technique for detecting steganographic images, but by embedding them into a convolutional neural network, [11] is able to achieve even better detection performance against a wide variety of steganography algorithms. We explore the performance of this model on various tasks in more detail in the following section.

	Test Performance		
Base Datasets	**LSB**	**HiDDeN**	**SteganoGAN**
LSB	0.914 ± 0.130	0.783 ± 0.067	0.629 ± 0.050
LSB + 1 SteganoGAN	0.929 ± 0.007	0.834 ± 0.019	0.815 ± 0.028
LSB + 2 SteganoGAN	0.940 ± 0.002	0.868 ± 0.013	0.890 ± 0.015
LSB + 3 SteganoGAN	0.950 ± 0.003	0.893 ± 0.002	0.892 ± 0.009
LSB + 4 SteganoGAN	0.946 ± 0.003	0.868 ± 0.007	0.939 ± 0.003
LSB + 5 SteganoGAN	0.952 ± 0.013	0.894 ± 0.009	0.940 ± 0.002
LSB + 6 SteganoGAN	0.955 ± 0.005	0.891 ± 0.009	0.962 ± 0.003
LSB + 7 SteganoGAN	0.965 ± 0.002	0.930 ± 0.020	0.958 ± 0.003
LSB + 8 SteganoGAN	0.962 ± 0.006	0.891 ± 0.009	0.973 ± 0.005
LSB + 9 SteganoGAN	0.971 ± 0.003	0.925 ± 0.007	0.971 ± 0.008
LSB + 10 SteganoGAN	0.965 ± 0.006	0.911 ± 0.004	0.978 ± 0.003

Table 1: This table shows the detection performance for YeNet models trained on various subsets of our base training datasets. We report the performance on the test sets which contain examples from LSB, HiDDeN, and SteganoGAN.

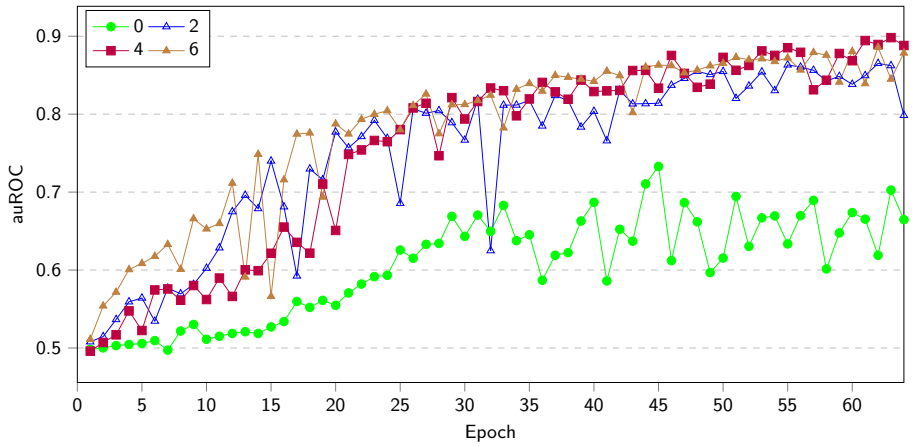

Fig. 4: This figure shows the area under the ROC curve on the SteganoGAN test set over time for models trained on different numbers of model instances. We see that models trained on more instances generally outperform models trained on fewer instances.

4 Experiments

To support our experiments, we start by creating two sets of cover images: *base* and *large*. The *base* set is smaller and consists of 1,000 randomly selected images, common objects and scenes from the COCO dataset [9]. The size of this *base* set is on par with other steganalysis datasets such as the datasets used in the "break our steganographic system" challenge [2]. The *large* set is also randomly selected

from the COCO dataset but contains 10,000 images. Both of these datasets are partitioned into a train and test partition which contain 70% and 30% of the images, respectively.

Train. To create our training datasets, we take the images in the *base* train partition and generate the corresponding steganographic images using the least significant bits algorithm as well as 10 different instances of SteganoGAN, giving us a total of 11 different *base* training datasets corresponding to different steganography algorithms. We use an identical procedure to generate the *large* training datasets.

Test. To create our test datasets, we take the images in the *base* test partition and generate the corresponding steganographic images using the least significant bit algorithm, a *secret* instance of SteganoGAN, and a *secret* instance of HiDDeN. Once again, we use an identical procedure to generate the *large* test datasets. This procedure ensures that (1) the LSB test set contains images that the model has never seen before, (2) the SteganoGAN test set contains images generated by a *specific set of neural network weights* that the model has never seen before, and (3) the HiDDeN test set contains images generated by a *entire class of steganography algorithms* that the model has never seen before.

Optimization. To train the YeNet model, we use the Adam optimizer with a batch size of 32 and an initial learning rate of 0.001, and decay the learning rate when the loss plateaus for 10 epochs. We train the model for 64 epochs and report the average of the area under the receiver operating characteristic over three training runs for each of the test sets. By varying the number of SteganoGAN instances used to train the YeNet model, we can measure the resulting change in performance on each of the test datasets.

5 Analysis

The results on the *base* datasets are shown in Table 1. We immediately see that a model trained solely on LSB images is not effective at detecting HiDDeN or SteganoGAN images. However, we also observe that as we add examples of steganographic images generated by different instances of SteganoGAN, the test performance increases across the board. Not only does the model get better at detecting steganographic images generated by a secret *instance* of SteganoGAN that it has never seen before, but it also gets better at detecting steganographic images generated by HiDDeN, a secret *class* of steganography algorithms that it has never seen before.

To further establish the robustness of our results, we repeat these experiments with the *large* datasets. The models trained on this dataset achieve dramatically higher detection accuracy than the models trained on the *base* dataset. However, we still observe similar trends: as we add more instances of SteganoGAN, our model becomes better at detecting images generated by previously unseen

	Test Performance		
Large Datasets	**LSB**	**HiDDeN**	**SteganoGAN**
LSB	0.989 ± 0.002	0.895 ± 0.013	0.812 ± 0.009
LSB + 1 SteganoGAN	0.983 ± 0.004	0.952 ± 0.004	0.976 ± 0.003
LSB + 2 SteganoGAN	0.989 ± 0.001	0.967 ± 0.001	0.984 ± 0.003
LSB + 3 SteganoGAN	0.988 ± 0.001	0.964 ± 0.004	0.989 ± 0.001
LSB + 4 SteganoGAN	0.989 ± 0.002	0.970 ± 0.001	0.991 ± 0.001
LSB + 5 SteganoGAN	0.990 ± 0.002	0.962 ± 0.002	0.993 ± 0.001

Table 2: This table shows the detection performance for YeNet models trained on various subsets of our *large* training datasets. We report the performance on the test sets, which contain examples from LSB, HiDDeN, and SteganoGAN.

steganography algorithms. We find that despite not providing a single example of a steganographic image generated by HiDDeN, we are able to detect them with an auROC of 0.971. These results suggests that using a diverse set of adversarially generated examples to train a steganalysis tool is a promising strategy for enabling steganalysis models to generalize well to steganography algorithms that it has never seen before.

6 Conclusion

In this paper, we explored the relationship between deep learning-based steganography algorithms and steganalysis techniques. We examined the impact of using multiple instances of the SteganoGAN steganography algorithm to train the YeNet steganalysis tool and found significant improvements on all of our test sets. Finally, we found evidence to suggest that by using a diverse set of adversarially generated examples as our test set, we can train steganalysis models which generalize well and are able to achieve high detection rates on steganography algorithms that the model has not seen before.

References

1. Baluja, S.: Hiding images in plain sight: Deep steganography. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 2069–2079. Curran Associates, Inc. (2017)
2. Bas, P., Filler, T., Pevnỳ, T.: "break our steganographic system": the ins and outs of organizing boss. In: International workshop on information hiding. pp. 59–70. Springer (2011)
3. Boehm, B.: StegExpose - A tool for detecting LSB steganography. CoRR **abs/1410.6656** (2014)
4. Dumitrescu, S., Wu, X., Memon, N.: On steganalysis of random lsb embedding in continuous-tone images. vol. 3, pp. 641 – 644 vol.3 (07 2002). https://doi.org/10.1109/ICIP.2002.1039052

5. Dumitrescu, S., Wu, X., Wang, Z.: Detection of LSB steganography via sample pair analysis. In: Information Hiding. pp. 355–372 (2003)
6. Fridrich, J., Kodovsky, J.: Rich models for steganalysis of digital images. IEEE Transactions on Information Forensics and Security **7**(3), 868–882 (June 2012). https://doi.org/10.1109/TIFS.2012.2190402
7. Fridrich, J., Goljan, M., Du, R.: Reliable detection of lsb steganography in color and grayscale images. In: Proc. of the 2001 Workshop on Multimedia and Security: New Challenges. pp. 27–30. ACM (2001). https://doi.org/10.1145/1232454.1232466
8. Hayes, J., Danezis, G.: Generating steganographic images via adversarial training. In: NIPS (2017)
9. Lin, T., Maire, M., Belongie, S.J., Bourdev, L.D., Girshick, R.B., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. CoRR **abs/1405.0312** (2014)
10. Wu, P., Yang, Y., Li, X.: Stegnet: Mega image steganography capacity with deep convolutional network. Future Internet **10**, 54 (06 2018). https://doi.org/10.3390/fi10060054
11. Ye, J., Ni, J., Yi, Y.: Deep learning hierarchical representations for image steganalysis. IEEE Trans. on Information Forensics and Security **12**(11), 2545–2557 (Nov 2017). https://doi.org/10.1109/TIFS.2017.2710946
12. Zhang, K.A., Cuesta-Infante, A., Xu, L., Veeramachaneni, K.: SteganoGAN: High capacity image steganography with gans. CoRR **abs/1901.03892** (2019), http://arxiv.org/abs/1901.03892
13. Zhu, J., Kaplan, R., Johnson, J., Fei-Fei, L.: HiDDeN: Hiding data with deep networks. CoRR **abs/1807.09937** (2018)