SteganoGAN: High Capacity Image Steganography with GANs

Kevin A. Zhang,[1] Alfredo Cuesta-Infante,[2] Lei Xu,[1] Kalyan Veeramachaneni[1]

[1] MIT, Cambridge, MA - 02139, USA
`kevz,leix,kalyanv@mit.edu`
[2] Univ. Rey Juan Carlos, Spain
`alfredo.cuesta@urjc.es`

Abstract

Image steganography is a procedure for hiding messages inside pictures. While other techniques such as cryptography aim to prevent adversaries from reading the secret message, steganography aims to hide the presence of the message itself. In this paper, we propose a novel technique for hiding arbitrary binary data in images using generative adversarial networks which allow us to optimize the perceptual quality of the images produced by our model. We show that our approach achieves state-of-the-art payloads of 4.4 bits per pixel, evades detection by steganalysis tools, and is effective on images from multiple datasets. To enable fair comparisons, we have released an open source library that is available online at: `https://github.com/DAI-Lab/SteganoGAN`.

1. Introduction

The goal of image steganography is to hide a secret message inside an image. In a typical scenario, the sender hides a secret message inside a cover image and transmits it to the receiver, who recovers the message. Even if the image is intercepted, no one besides the sender and receiver should be able to detect the presence of a message.

Traditional approaches to image steganography are only effective up to a relative payload of around 0.4 bits per pixel (Pevný et al., 2010). Beyond that point, they tend to introduce artifacts that can be easily detected by automated steganalysis tools and, in extreme cases, by the human eye. With the advent of deep learning in the past decade, a new class of image steganography approaches is emerging (Hayes & Danezis, 2017; Baluja, 2017; Zhu et al., 2018). These approaches use neural networks as either a component in a traditional algorithm (e.g. using deep learning to

---

Correspondence to: Kevin A. Zhang <kevz@mit.edu>.

Preprint.

identify spatial locations suitable for embedding data), or as an end-to-end solution, which takes in a cover image and a secret message and combines them into a steganographic image.

These attempts have proved that deep learning can be used for practical end-to-end image steganography, and have achieved embedding rates competitive with those accomplished through traditional techniques (Pevný et al., 2010). However, they are also more limited than their traditional counterparts: they often impose special constraints on the size of the cover image (for example, (Hayes & Danezis, 2017) requires the cover images to be 32 x 32); they attempt to embed images inside images and not arbitrary messages or bit vectors; and finally, they do not explore the limits of how much information can be hidden successfully. We provide the reader a detailed analysis of these methods in Section 7.

To address these limitations, we propose STEGANOGAN, a novel end-to-end model for image steganography that builds on recent advances in deep learning. We use dense connections which mitigate the vanishing gradient problem and have been shown to improve performance (Huang et al., 2017). In addition, we use multiple loss functions within an adversarial training framework to optimize our encoder, decoder, and critic networks simultaneously. We find that our approach successfully embeds arbitrary data into cover images drawn from a variety of natural scenes and achieves state-of-the-art embedding rates of 4.4 bits per pixel while evading standard detection tools. Figure 1 presents some example images that demonstrate the effectiveness of STEGANOGAN. The left-most figure is the original cover image without any secret messages. The next four figures contain approximately 1, 2, 3, and 4 bits per pixel worth of secret data, respectively, without producing any visible artifacts.

Our contributions through this paper are:

- We present a novel approach that uses adversarial training to solve the steganography task and achieves a relative payload of 4.4 bits per pixel which is 10x higher

Figure 1. A randomly selected cover image (left) and the corresponding steganographic images generated by STEGANOGAN at approximately 1, 2, 3, and 4 bits per pixel.

than competing deep learning-based approaches with similar peak signal to noise ratios.

– We propose a new metric for evaluating the *capacity* of deep learning-based steganography algorithms, which enables comparisons against traditional approaches.

– We evaluate our approach by measuring its ability to evade traditional steganalysis tools which are designed to detect whether an image is steganographic or not. Even when we encode > 4 bits per pixel into the image, most traditional steganalysis tools still only achieve a detection auROC of < 0.6.

– We also evaluate our approach by measuring its ability to evade deep learning-based steganalysis tools. We train a state-of-the-art model for automatic steganalysis proposed by (Ye et al., 2017) on samples generated by our model. If we require our model to produce steganographic images such that the detection rate is at most 0.8 auROC, we find that our model can still hide up to 2 bits per pixel.

– We are releasing a fully-maintained open-source library called STEGANOGAN[1], including datasets and pre-trained models, which will be used to evaluate deep learning based steganography techniques.

The rest of the paper is organized as follows. Section 2 briefly describes our motivation for building a better image steganography system. Section 3 presents STEGANOGAN and describes our model architecture. Section 4 describes our metrics for evaluating model performance. Section 5 contains our experiments for several variants of our model. Section 6 explores the effectiveness of our model at avoiding detection by automated steganalysis tools. Section 7 details related work in the generation of steganographic images.

2. Motivation

There are several reasons to use steganography instead of (or in addition to) cryptography when communicating a

secret message between two actors. First, the information contained in a cryptogram is accessible to anyone who has the private key, which poses a challenge in countries where private key disclosure is required by law. Furthermore, the very existence of a cryptogram reveals the presence of a message, which can invite attackers. These problems with plain cryptography exist in security, intelligence services, and a variety of other disciplines (Conway, 2003).

For many of these fields, steganography offers a promising alternative. For example, in medicine, steganography can be used to hide private patient information in images such as X-rays or MRIs (Srinivasan et al., 2004) as well as biometric data (Douglas et al., 2018). In the media sphere, steganography can be used to embed copyright data (Maheswari & Hemanth, 2015) and allow content access control systems to store and distribute digital works over the Internet (Kawaguchi et al., 2007). In each of these situations, it is important to embed as much information as possible, and for that information to be both *undetectable* and *lossless* to ensure the data can be recovered by the recipient. Most work in the area of steganography, including the methods described in this paper, targets these two goals. We propose a new class of models for image steganography that achieves both these goals.

3. SteganoGAN

In this section, we introduce our notation, present the model architecture, and describe the training process. At a high level, steganography requires just two operations: *encoding* and *decoding*. The *encoding* operation takes a cover image and a binary message, and creates a steganographic image. The *decoding* operation takes the steganographic image and recovers the binary message.

3.1. Notation

We have C and S as the cover image and the steganographic image respectively, both of which are RGB color images and have the same resolution $W \times H$; let $M \in \{0,1\}^{D \times W \times H}$ be the binary message that is to be hidden in C. Note that D

---

[1] https://github.com/DAI-Lab/SteganoGAN

is the upper-bound on the relative payload; the actual relative payload is the number of bits that can reliably decoded which is given by $(1 - 2p)D$, where $p \in [0, 1]$ is the error rate. The actual relative payload is discussed in more detail in Section 4.

The cover image C is sampled from the probability distribution of all natural images $\mathbb{P}_C$. The steganographic image S is then generated by a learned encoder $\mathcal{E}(C, M)$. The secret message $\hat{M}$ is then extracted by a learned decoder $\mathcal{D}(S)$. The optimization task, given a fixed message distribution, is to train the encoder $\mathcal{E}$ and the decoder $\mathcal{D}$ to minimize (1) the decoding error rate p and (2) the distance between natural and steganographic image distributions $dis(\mathbb{P}_C, \mathbb{P}_S)$. Therefore, to optimize the encoder and the decoder, we also need to train a critic network $\mathcal{C}(\cdot)$ to estimate $dis(\mathbb{P}_C, \mathbb{P}_S)$.

Let $X \in \mathbb{R}^{D \times W \times H}$ and $Y \in \mathbb{R}^{D' \times W \times H}$ be two tensors of the same width and height but potentially different depth, D and D'; then, let $\mathtt{Cat} : (X, Y) \rightarrow \Phi \in \mathbb{R}^{(D+D') \times W \times H}$ be the concatenation of the two tensors along the depth axis.

Let $\mathtt{Conv}_{D \rightarrow D'} : X \in \mathbb{R}^{D \times W \times H} \rightarrow \Phi \in \mathbb{R}^{D' \times W \times H}$ be a convolutional block that maps an input tensor X into a feature map Φ of the same width and height but potentially different depth. This convolutional block consists of a convolutional layer with kernel size 3, stride 1 and padding 'same', followed by a leaky ReLU activation function and batch normalization. The activation function and batch normalization operations are omitted if the convolutional block is the last block in the network.

Let $\mathtt{Mean} : X \in \mathbb{R}^{D \times W \times H} \rightarrow \mathbb{R}^D$ represent the adaptive mean spatial pooling operation which computes the average of the $W \times H$ values in each feature map of tensor X.

3.2. Architecture

In this paper, we present STEGANOGAN, a generative adversarial network for hiding an arbitrary bit vector in a cover image. Our proposed architecture, shown in Figure 2, consists of three modules: (1) an Encoder that takes a cover image and a data tensor, or message, and produces a steganographic image (Section 3.2.1); (2) a Decoder that takes the steganographic image and attempts to recover the data tensor (Section 3.2.2), and (3) a Critic that evaluates the quality of the cover and steganographic images (Section 3.2.3).

3.2.1. ENCODER

The encoder network takes a cover image C and a message $M \in \{0, 1\}^{D \times W \times H}$. Hence M is a binary data tensor of shape $D \times W \times H$ where D is the number of bits that we will attempt to hide in each pixel of the cover image.

We explore three variants of the encoder architecture with different connectivity patterns. All the variants start by

applying the following two operations:

1. Processing the cover image C with a convolutional block to obtain the tensor a given by

$$a = \mathtt{Conv}_{3 \rightarrow 32}(C) \qquad (1)$$

2. Concatenating the message M to a and then processing the result with a convolutional block to obtain the tensor b:

$$b = \mathtt{Conv}_{32+D \rightarrow 32}(\mathtt{Cat}(a, M)) \qquad (2)$$

Basic: We sequentially apply two convolutional blocks to tensor b and generate the steganographic image as shown in Figure 2b. Formally:

$$\mathcal{E}_b(C, M) = \mathtt{Conv}_{32 \rightarrow 3}(\mathtt{Conv}_{32 \rightarrow 32}(b)), \qquad (3)$$

This approach is similar to that in (Baluja, 2017) as the steganographic image is simply the output of the last convolutional block.

Residual: The use of residual connections has been shown to improve model stability and convergence (He et al., 2016) so we hypothesize that its use will improve the quality of the steganographic image. To this end we modify the basic encoder by adding the cover image C to its output so that the encoder learns to produce a residual image as shown in Figure 2c. Formally,

$$\mathcal{E}_r(C, M) = C + \mathcal{E}_b(C, M), \qquad (4)$$

Dense: In the dense variant, we introduce additional connections between the convolutional blocks so that the feature maps generated by the earlier blocks are concatenated to the feature maps generated by later blocks as shown in Figure 2d. This connectivity pattern is inspired by the DenseNet (Huang et al., 2017) architecture which has been shown to encourage feature reuse and mitigate the vanishing gradient problem. Therefore, we hypothesize that the use of dense connections will improve the embedding rate. It can be formally expressed as follows

$$\begin{cases} c = \mathtt{Conv}_{64+D \rightarrow 32}(\mathtt{Cat}(a, b, M)) \\ d = \mathtt{Conv}_{96+D \rightarrow 3}(\mathtt{Cat}(a, b, c, M)) \\ \mathcal{E}_d(C, M) = C + d \end{cases} \qquad (5)$$

Finally, the output of each variant is a steganographic image $S = \mathcal{E}_{\{b,r,d\}}(C, M)$ that has the same resolution and depth than the cover image C.
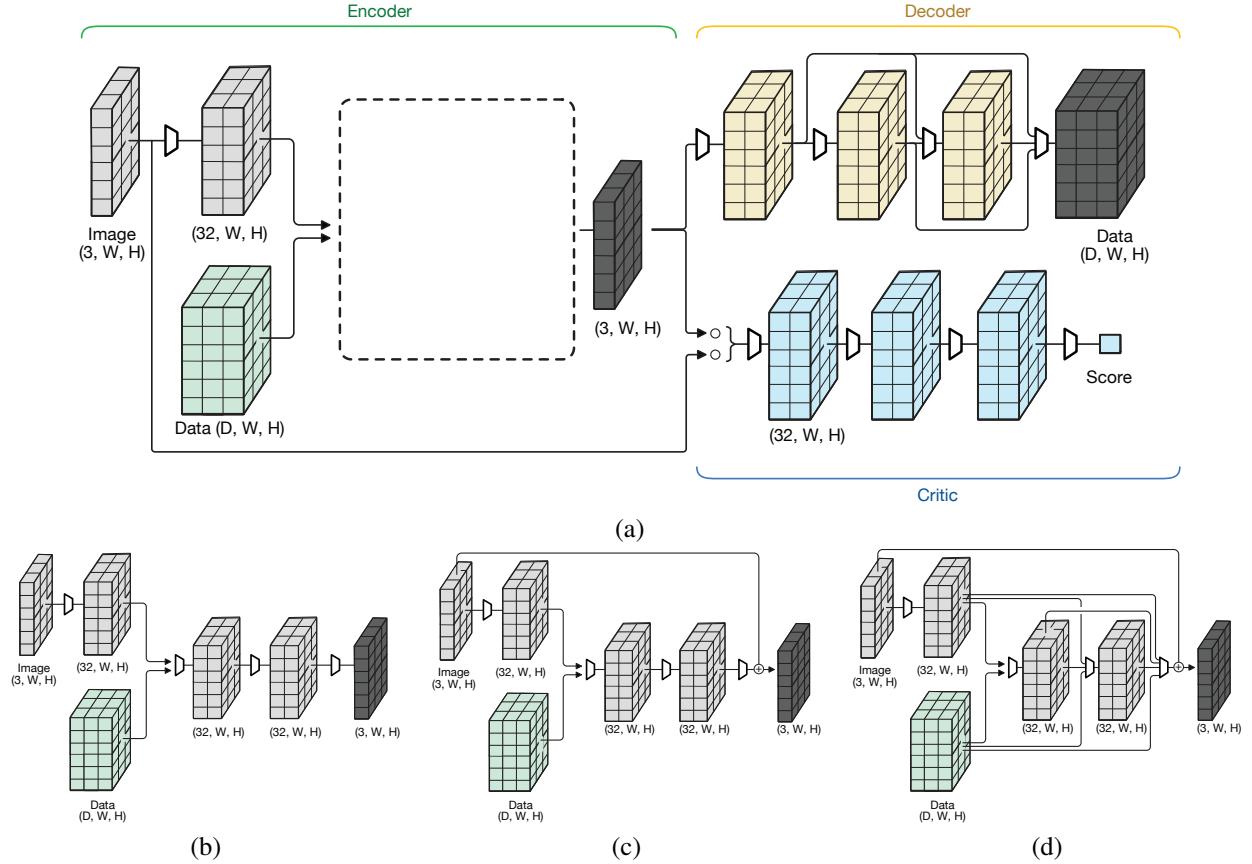
Figure 2. (a) The model architecture with the Encoder, Decoder, and Critic. The blank rectangle representing the Encoder can be any of the following: (b) Basic encoder, (c) Residual encoder and (d) Dense encoder. The trapezoids represent convolutional blocks, two or more arrows merging represent concatenation operations, and the curly bracket represents a batching operation.

3.2.2. DECODER

The decoder network takes the steganographic image S produced by the encoder. Formally it can be expressed as:

$$\begin{cases} a = \texttt{Conv}_{3\to32}(S) \\ b = \texttt{Conv}_{32\to32}(a) \\ c = \texttt{Conv}_{64\to32}(\texttt{Cat}(a,b)) \\ \mathcal{D}(S) = \texttt{Conv}_{96\to D}(\texttt{Cat}(a,b,c)) \end{cases} \quad (6)$$

The decoder produces $\hat{M} = \mathcal{D}_d(S)$; in other words it attempts to recover the data tensor M.

3.2.3. CRITIC

To provide feedback on the performance of our encoder and generate more realistic images, we introduce an adversarial Critic. The critic network consists of three convolutional blocks followed by a convolutional layer with one output channel. To produce the scalar score, we apply adaptive mean pooling to the output of the convolutional layer.

$$\begin{cases} a = \texttt{Conv}_{32\to32}(\texttt{Conv}_{32\to32}(\texttt{Conv}_{3\to32}(S))) \\ \mathcal{C}(S) = \texttt{Mean}(\texttt{Conv}_{32\to1}((a)) \end{cases} \quad (7)$$

3.3. Training

We iteratively optimize the encoder-decoder network and the critic network. To optimize the encoder-decoder network, we jointly optimize three losses: (1) the decoding accuracy using the cross-entropy loss

$$\mathcal{L}_d = \mathbb{E}_{X \sim \mathbb{P}_C} \text{CrossEntropy}(\mathcal{D}(\mathcal{E}(X, M)), M) \quad (8)$$

(2) the similarity between steganographic image and the cover image using mean square error

$$\mathcal{L}_s = \mathbb{E}_{X \sim \mathbb{P}_C} \frac{1}{3 \times W \times H} ||X - \mathcal{E}(X, M)||_2^2 \quad (9)$$

(3) and the realness of the steganographic image using the critic network

$$\mathcal{L}_r = \mathbb{E}_{X \sim \mathbb{P}_C} \mathcal{C}(\mathcal{E}(X, M)) \quad (10)$$

The training objective is to

$$\text{minimize } \mathcal{L}_d + \mathcal{L}_s + \mathcal{L}_r. \qquad (11)$$

To train the critic network, we minimize the Wasserstein loss

$$\begin{aligned}
\mathcal{L}_c =& \mathbb{E}_{X \sim \mathbb{P}_C} \mathcal{C}(X) \\
& - \mathbb{E}_{X \sim \mathbb{P}_C} \mathcal{C}(\mathcal{E}(X, M))
\end{aligned} \qquad (12)$$

During every iteration, we match each cover image C with a data tensor M, which consists of a randomly generated sequence of $D \times W \times H$ bits sampled from a Bernoulli distribution $M \sim Ber(0.5)$. In addition, we apply standard data augmentation procedures including horizontal flipping and random cropping to cover image C in our pre-processing pipeline. We use the Adam optimizer with learning rate $1e^{-4}$, clip our gradient norm to 0.25, clip the critic weights to $[-0.1, 0.1]$, and train for 32 epochs.

4. Evaluation Metrics

Steganography algorithms are evaluated along three axes: the amount of data that can be hidden in an image, a.k.a *capacity*, the similarity between the cover and steganography image, a.k.a *distortion*, and the ability to avoid detection by steganalysis tools, a.k.a *secrecy*. This section describes some metrics for evaluating the performance of our model along these axes.

Reed Solomon Bits Per Pixel: Measuring the effective number of bits that can be conveyed per pixel is non-trivial in our setup since the ability to recover a hidden bit is heavily dependent on the model and the cover image, as well as the message itself.

To model this situation, suppose that a given model incorrectly decodes a bit with probability p. It is tempting to just multiply the number of bits in the data tensor by the accuracy $1 - p$ and report that value as the relative payload. Unfortunately, that value is actually meaningless – it allows you to estimate the number of bits that have been correctly decoded, but does not provide a mechanism for recovering from errors or even identifying which bits are correct.

Therefore, to get an accurate estimate of the relative payload of our technique, we turn to Reed-Solomon codes. Reed-Solomon error-correcting codes are a subset of linear block codes which offer the following guarantee: Given a message of length k, the code can generate a message of length n where $n \geq k$ such that it can recover from $\frac{n-k}{2}$ errors (Reed & Solomon, 1960). This implies that given a steganography algorithm which, on average, returns an incorrect bit with probability p, we would want the number of incorrect bits

to be less than or equal to the number of bits we can correct:

$$p \cdot n \leq \frac{n - k}{2} \qquad (13)$$

The ratio k/n represents the average number of bits of "real" data we can transmit for each bit of "message" data; then, from (13), it follows that the ratio is less than or equal to $1 - 2p$. As a result, we can measure the relative payload of our steganographic technique by multiplying the number of bits we attempt to hide in each pixel by the ratio to obtain the "real" number of bits that is transmitted and recovered.

We refer to this metric as Reed-Solomon bits-per-pixel (RS-BPP), and note that it can be directly compared against traditional steganographic techniques since it represents the average number of bits that can be reliably transmitted in an image divided by the size of the image.

Peak Signal to Noise Ratio: In addition to measuring the relative payload, we also need to measure the quality of the steganographic image. One widely-used metric for measuring image quality is the peak signal-to-noise ratio (PSNR). This metric is designed to measure image distortions and has been shown to be correlated with mean opinion scores produced by human experts (Wang et al., 2004).

Given two images X and Y of size (W, H) and a scaling factor `sc` which represents the maximum possible difference in the numerical representation of each pixel[2], the PSNR is defined as a function of the mean squared error (MSE):

$$\text{MSE} = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} (X_{i,j} - Y_{i,j})^2, \qquad (14)$$

$$\text{PSNR} = 20 \cdot \log_{10}(\texttt{sc}) - 10 \cdot \log_{10}(\text{MSE}) \qquad (15)$$

Although PSNR is widely used to evaluate the distortion produced by steganography algorithms, (Almohammad & Ghinea, 2010) suggests that it may not be ideal for comparisons across different types of steganography algorithms. Therefore, we introduce another metric to help us evaluate image quality: the structural similarity index.

Structural Similarity Index: In our experiments, we also report the structural similarity index (SSIM) between the cover image and the steganographic image. SSIM is widely used in the broadcast industry to measure image and video quality (Wang et al., 2004). Given two images X and Y, the SSIM can be computed using the means, μ_X and μ_Y,

---

[2]For example, if the images are represented as floating point numbers in $[-1.0, 1.0]$, then $\texttt{sc} = 2.0$ since the maximum difference between two pixels is achieved when one is 1.0 and the other is -1.0.

Figure 3. Randomly selected pairs of cover (left) and steganographic (right) images from the COCO dataset which embeds random binary data at the maximum payload of 4.4 bits-per-pixel.

variances, σ_X^2 and σ_Y^2, and covariance σ_{XY}^2 of the images as shown below:

$$\text{SSIM} = \frac{(2\mu_X\mu_Y + k_1 R)(2\sigma_{XY} + k_2 R)}{(\mu_X^2 + \mu_Y^2 + k_1 R)(\sigma_X^2 + \sigma_Y^2 + k_2 R)} \quad (16)$$

The default configuration for SSIM uses $k_1 = 0.01$ and $k_2 = 0.03$ and returns values in the range $[-1.0, 1.0]$ where 1.0 indicates the images are identical.

5. Results and Analysis

We use the Div2k (Agustsson & Timofte, 2017) and COCO (Lin et al., 2014) datasets to train and evaluate our model. We experiment with each of the three model variants discussed in Section 3 and train them with 6 different data depths $D \in \{1, 2, ..., 6\}$. The data depth D represents the "target" bits per pixel so the randomly generated data tensor has shape D x W x H.

We use the default train/test split proposed by the creators of the Div2K and COCO data sets in our experiments, and we report the average RS-BPP, PSNR, and SSIM on the test set in Table 1. Our models are trained on GeForce GTX 1080 GPUs. The wall clock time per epoch is approximately 10 minutes for Div2K and 2 hours for COCO.

After training our model, we compute the expected accuracy on a held-out test set and adjust it using the Reed-Solomon

coding scheme discussed in Section 4 to produce our bits-per-pixel metric, shown in Table 1 under RS-BPP. We publicly released the pre-trained models for all the experiments shown in this table on AWS S3[3].

The results from our experiments are shown in Table 1 – each of the metrics is computed on a held-out test set of images that is not shown to the model during training. Note that there is an unavoidable tradeoff between the relative payload and the image quality measures; assuming we are already on the Pareto frontier, an increased relative payload would inevitably result in a decreased similarity.

We immediately observe that all variants of our model perform better on the COCO dataset than the Div2K dataset. This can be attributed to differences in the type of content photographed in the two datasets. Images from the Div2K dataset tend to contain open scenery, while images from the COCO dataset tend to be more cluttered and contain multiple objects, providing more surfaces and textures for our model to successfully embed data.

In addition, we note that our dense variant shows the best performance on both relative payload and image quality, followed closely by the residual variant which shows comparable image quality but a lower relative payload. The basic variant offers the worst performance across all metrics, achieving relative payloads and image quality scores that are 15-25% lower than the dense variant.

---
[3]http://steganogan.s3.amazonaws.com/

Dataset	D	Accuracy			RS-BPP			PSNR			SSIM		
		Basic	Resid.	Dense	Basic	Resid	Dense	Basic	Resid.	Dense	Basic	Resid.	Dense
Div2K	1	0.95	0.99	1.00	0.91	0.99	0.99	24.52	41.68	41.60	0.70	0.96	0.95
	2	0.91	0.98	0.99	1.65	1.92	1.96	24.62	38.25	39.62	0.67	0.90	0.92
	3	0.82	0.92	0.94	1.92	2.52	2.63	25.03	36.67	36.52	0.69	0.85	0.85
	4	0.75	0.82	0.82	1.98	2.52	2.53	24.45	37.86	37.49	0.69	0.88	0.88
	5	0.69	0.74	0.75	1.86	2.39	2.50	24.90	39.45	38.65	0.70	0.90	0.90
	6	0.67	0.69	0.70	2.04	2.32	2.44	24.72	39.53	38.94	0.70	0.91	0.90
COCO	1	0.98	0.99	0.99	0.96	0.99	0.99	31.21	41.71	42.09	0.87	0.98	0.98
	2	0.97	0.99	0.99	1.88	1.97	1.97	31.56	39.00	39.08	0.86	0.96	0.95
	3	0.94	0.97	0.98	2.67	2.85	2.87	30.16	37.38	36.93	0.83	0.93	0.92
	4	0.87	0.95	0.95	2.99	3.60	3.61	31.12	36.98	36.94	0.83	0.92	0.92
	5	0.84	0.90	0.92	3.43	3.99	4.24	29.73	36.69	36.61	0.80	0.90	0.91
	6	0.78	0.84	0.87	3.34	4.07	4.40	31.42	36.75	36.33	0.84	0.89	0.88

Table 1. The relative payload and image quality metrics for each dataset and model variant. The Dense model variant offers the best performance across all metrics in almost all experiments.
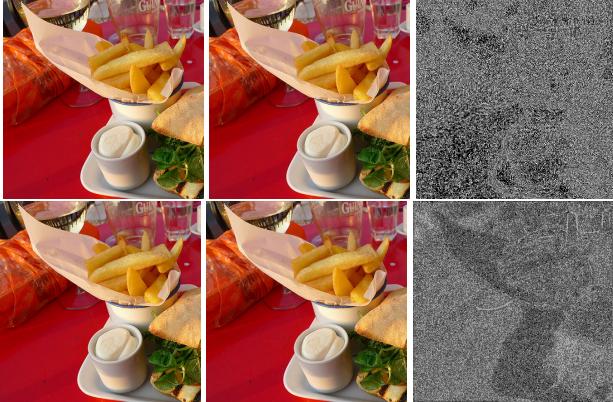

Figure 4. A randomly selected pair of cover (left) and steganographic (right) images and the differences between them. The top row shows the output from a simple least-significant-bit steganography algorithm (Johnson & C. Katzenbeisser, 1999) while the bottom row shows the output from STEGANOGAN with 4.4 bpp. Note that STEGANOGAN is able to adapt to the image content.

Finally, we remark that despite the increased relative payload, the image similarity as measured by the average peak signal to noise ratio between the cover image and the steganographic images produced by the Dense models are comparable to that presented in (Zhu et al., 2018).

6. Detecting Steganographic Images

Steganography techniques are also typically evaluated by their ability to evade detection by steganalysis tools. In this section, we experiment with two open source steganalysis algorithms and measure our model's ability to generate undetectable steganographic images.

6.1. Statistical Steganalysis

We use a popular open-source steganalysis tool called StegExpose (Boehm, 2014) which combines several existing steganalysis techniques including Sample Pairs (Dumitrescu et al., 2003), RS Analysis (Fridrich et al., 2001), Chi Squared Attack (Westfeld & Pfitzmann, 2000), and Primary Sets (Dumitrescu et al., 2002). To measure the effectiveness of our method at evading detection by these techniques, we randomly select 1,000 cover images from the test set, generating the corresponding steganographic images using our Dense architecture with data depth 6, and examine the results using StegExpose.

The receiver operating characteristic curve for our Dense model is shown in Figure 5 and we note that the StegExpose tool is only slightly more effective than random guessing with an area under the ROC curve of 0.59, even for payloads of up to 4.4 bits per pixel. This shows that our model can successfully evade standard steganalysis tools, meeting the minimum requirement for being a viable steganography algorithm.

6.2. Neural Steganalysis

Recent studies have shown promising results in detecting steganographic images using deep learning based approaches (Ye et al., 2017). Therefore, we proceed to examine whether our model can evade deep learning-based steganalysis tools. We use the model proposed by Ye et al., 2017 in (Ye et al., 2017) for steganalysis, with a slight modification to enable support of color images, and train it to detect steganographic images generated by STEGANOGAN.

In a typical scenario, the party that is attempting to detect steganographic images has access to the algorithm used to
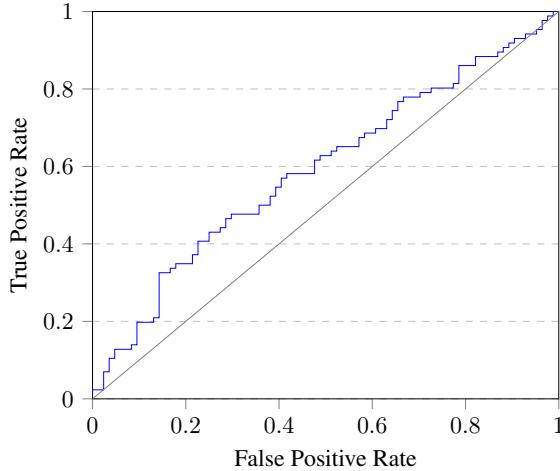
Figure 5. The receiver operating characteristic (ROC) curve produced by the StegExpose library for a set of 1000 steganographic images generated using the Dense architecture with a data depth of 6. The StegExpose library includes multiple steganalysis tools including `SamplePairs` (Dumitrescu et al., 2003), `RSAnalysis` (Fridrich et al., 2001), `ChiSquaredAttack` (Westfeld & Pfitzmann, 2000), and `PrimarySets` (Dumitrescu et al., 2002). The tool achieves an auROC of 0.59.

Figure 6. This plot shows the performance of the steganography detector on a held-out test set. The x-axis indicates the number of different STEGANOGAN instances that were used, while the y-axis indicates the area under the ROC curve.

create them - in our case, an instance of STEGANOGAN which is parameterized by the pretrained model weights but not the exact model. Using the Dense architecture from Section 3.2.1 and the COCO dataset, we attempt to replicate this scenario with the following experimental setup:

1. We train N instances of the Dense STEGANOGAN architecture with different random seeds.

2. For each of these trained models, we generate a set of 1,000 steganographic images.

3. Holding out the images generated by the Nth model as a test set, we train the steganalysis model proposed in (Ye et al., 2017) on increasing subsets of the remaining images $\{1\}, \{1, 2\}, \{1, 2, 3\}, ..., \{1, 2, 3, ..., N-1\}$.

4. We repeat each experiment 3 times and report the average area under the receiver operating curve in Figure 6.

This emulates a realistic setting - the party creating the automatic detection model will not have access to the specific STEGANOGAN model in use, but may have access to the software used to train the models. Therefore, we pose the following question: If the external party does not know the specific model weights but does know the algorithm for generating models, can they detect steganographic images generated by STEGANOGAN?

Figure 6 shows the performance of our detector for various relative payloads and training set sizes. First, we note that
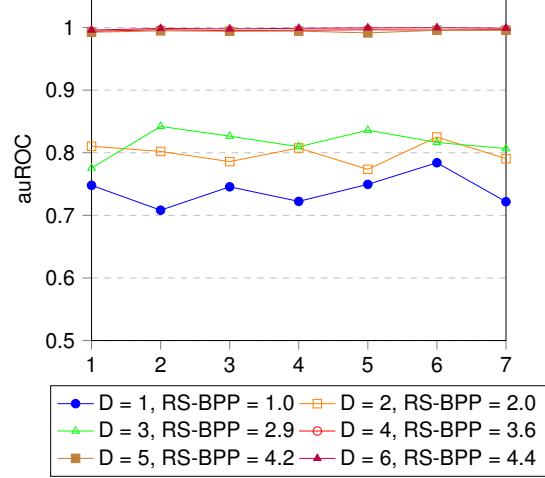
the detector performance, as measured by the area under the receiver operating characteristic (auROC), increases as we increase the number of bits-per-pixel encoded in the image. In addition, we highlight the fact there is no clear trend in the area under the ROC curve as we increase the number of STEGANOGAN models used for training. This suggests that the external party will have a difficult time building a model which can detect steganographic images generated by STEGANOGAN without knowing the exact model parameters.

Finally, we compare the detection error for images generated by STEGANOGAN against those reported by (Ye et al., 2017) on images generated by three state-of-the-art steganography algorithms: WOW (Holub & Fridrich, 2012), S-UNIWARD (Holub et al., 2014), and HILL (Li et al., 2014). Note that these techniques are evaluated on different dataset and as such, the results are only approximate estimates of the actual relative payload achievable on a particular dataset. For a fixed detection error rate of 20%, we find that WOW is able to encode up to 0.3 bpp, S-UNIWARD is able to encode up to 0.4 bpp, HILL is able to encode up to 0.5 bpp, and STEGANOGAN is able to encode up to 2.0 bpp.

7. Related Work

In this section, we describe a few traditional approaches to image steganography and then discuss recent approaches developed using deep learning.

7.1. Traditional Approaches

A standard algorithm for image steganography is "Highly Undetectable steGO" (HUGO), a cost function-based algorithm which uses handcrafted features to measure the distortion introduced by modifying the pixel value at a particular location in the image. Given a set of N bits to be embedded, HUGO uses the distortion function to identify the top N pixels that can be modified while minimizing the total distortion across the image (Pevný et al., 2010).

Another approach is the JSteg algorithm, which is designed specifically for JPEG images. JPEG compression works by transforming the image into the frequency domain using the discrete cosine transform and removing high-frequency components, resulting in a smaller image file size. JSteg uses the same transformation into the frequency domain, but modifies the least significant bits of the frequency coefficients (Li et al., 2011).

7.2. Deep Learning for Steganography

Deep learning for image steganography has recently been explored in several studies, all showing promising results. These existing proposals range from training neural networks to integrate with and improve upon traditional steganography techniques (Tang et al., 2017) to complete end-to-end convolutional neural networks which use adversarial training to generate convincing steganographic images (Hayes & Danezis, 2017; Zhu et al., 2018).

Hiding images vs. arbitrary data: The first set of deep learning approaches to steganography were (Baluja, 2017; Wu et al., 2018). Both (Baluja, 2017) and (Wu et al., 2018) focus solely on taking a *secret image* and embedding it into a *cover image*. Because this task is fundamentally different from that of embedding arbitrary data, it is difficult to compare these results to those achieved by traditional steganography algorithms in terms of the relative payload.

Natural images such as those used in (Baluja, 2017) and (Wu et al., 2018) exhibit strong spatial correlations, and convolutional neural networks trained to hide images in images would take advantage of this property. Therefore, a model that is trained in such a manner cannot be applied to arbitrary data.

Adversarial training: The next set of approaches for image steganography are (Hayes & Danezis, 2017; Zhu et al., 2018) which make use of adversarial training techniques. The key differences between these approaches and our approach are the loss functions used to train the model, the architecture of the model, and how data is presented to the network.

The method proposed by (Hayes & Danezis, 2017) can only operate on images of a fixed size. Their approach involves flattening the image into a vector, concatenating the data vector to the image vector, and applying feedforward, reshaping, and convolutional layers. They use the mean squared error for the encoder, the cross entropy loss for the discriminator, and the mean squared error for the decoder. They report that image quality suffers greatly when attempting to increase the number of bits beyond 0.4 bits per pixel.

The method proposed by (Zhu et al., 2018) uses the same loss functions as (Hayes & Danezis, 2017) but makes changes to the model architecture. Specifically, they "*replicate the message spatially, and concatenate this message volume to the encoders intermediary representation.*" For example, in order to hide k bits in an $N \times N$ image, they would create a tensor of shape (k, N, N) where the data vector is replicated at each spatial location.

This design allows (Zhu et al., 2018) to handle arbitrary sized images but cannot effectively scale to higher relative payloads. For example, to achieve a relative payload of 1 bit per pixel in a typical image of size 360×480, they would need to manipulate a data tensor of size $(172800, 360, 480)$. Therefore, due to the excessive memory requirements, this model architecture cannot effectively scale to handle large relative payloads.

8. Conclusion

In this paper, we introduced a flexible new approach to image steganography which supports different-sized cover images and arbitrary binary data. Furthermore, we proposed a new metric for evaluating the performance of deep-learning based steganographic systems so that they can be directly compared against traditional steganography algorithms. We experiment with three variants of the STEGANOGAN architecture and demonstrate that our model achieves higher relative payloads than existing approaches while still evading detection.

Acknowledgements

References

Agustsson, E. and Timofte, R. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

Almohammad, A. and Ghinea, G. Stego image quality

and the reliability of psnr. In *2010 2nd International Conference on Image Processing Theory, Tools and Applications*, pp. 215–220, July 2010. doi: 10.1109/IPTA. 2010.5586786.

Baluja, S. Hiding images in plain sight: Deep steganography. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 2069–2079. Curran Associates, Inc., 2017.

Boehm, B. StegExpose - A tool for detecting LSB steganography. *CoRR*, abs/1410.6656, 2014.

Conway, M. Code wars: Steganography, signals intelligence, and terrorism. *Knowledge, Technology & Policy*, 16(2):45–62, Jun 2003. ISSN 1874-6314. doi: 10.1007/s12130-003-1026-4.

Douglas, M., Bailey, K., Leeney, M., and Curran, K. An overview of steganography techniques applied to the protection of biometric data. *Multimedia Tools and Applications*, 77(13):17333–17373, Jul 2018. ISSN 1573-7721. doi: 10.1007/s11042-017-5308-3.

Dumitrescu, S., Wu, X., and Memon, N. On steganalysis of random lsb embedding in continuous-tone images. volume 3, pp. 641 – 644 vol.3, 07 2002. doi: 10.1109/ICIP.2002.1039052.

Dumitrescu, S., Wu, X., and Wang, Z. Detection of LSB steganography via sample pair analysis. In *Information Hiding*, pp. 355–372, 2003. ISBN 978-3-540-36415-3.

Fridrich, J., Goljan, M., and Du, R. Reliable detection of lsb steganography in color and grayscale images. In *Proc. of the 2001 Workshop on Multimedia and Security: New Challenges*, MM&Sec '01, pp. 27–30. ACM, 2001. ISBN 1-58113-393-6. doi: 10.1145/1232454.1232466.

Hayes, J. and Danezis, G. Generating steganographic images via adversarial training. In *NIPS*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Holub, V. and Fridrich, J. Designing steganographic distortion using directional filters. 12 2012. doi: 10.1109/WIFS.2012.6412655.

Holub, V., Fridrich, J., and Denemark, T. Universal distortion function for steganography in an arbitrary domain. *EURASIP Journal on Information Security*, 2014 (1):1, Jan 2014. ISSN 1687-417X. doi: 10.1186/1687-417X-2014-1.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017.

Johnson, N. and C. Katzenbeisser, S. A survey of steganographic techniques. 01 1999.

Kawaguchi, E., Maeta, M., Noda, H., and Nozaki, K. A model of digital contents access control system using steganographic information hiding scheme. In *Proc. of the 18th Conf. on Information Modelling and Knowledge Bases*, pp. 50–61, 2007. ISBN 978-1-58603-710-9.

Li, B., He, J., Huang, J., and Shi, Y. A survey on image steganography and steganalysis. *Journal of Information Hiding and Multimedia Signal Processing*, 2011.

Li, B., Wang, M., Huang, J., and Li, X. A new cost function for spatial image steganography. In *2014 IEEE Int. Conf. on Image Processing (ICIP)*, pp. 4206–4210, Oct 2014. doi: 10.1109/ICIP.2014.7025854.

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

Maheswari, S. U. and Hemanth, D. J. Frequency domain qr code based image steganography using fresnelet transform. *AEU - International Journal of Electronics and Communications*, 69(2):539 – 544, 2015. ISSN 1434-8411. doi: https://doi.org/10.1016/j.aeue.2014.11.004.

Pevný, T., Filler, T., and Bas, P. Using high-dimensional image models to perform highly undetectable steganography. In *Information Hiding*, 2010.

Reed, I. S. and Solomon, G. Polynomial Codes Over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

Srinivasan, Y., Nutter, B., Mitra, S., Phillips, B., and Ferris, D. Secure transmission of medical records using high capacity steganography. In *Proc. of the 17th IEEE Symposium on Computer-Based Medical Systems*, pp. 122–127, June 2004. doi: 10.1109/CBMS.2004.1311702.

Tang, W., Tan, S., Li, B., and Huang, J. Automatic steganographic distortion learning using a generative adversarial network. *IEEE Signal Processing Letters*, 24(10):1547–1551, Oct 2017. ISSN 1070-9908. doi: 10.1109/LSP.2017.2745572.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. on Image Processing*, 13(4):600–612, April 2004. ISSN 1057-7149. doi: 10.1109/TIP.2003.819861.

Westfeld, A. and Pfitzmann, A. Attacks on steganographic systems. In *Information Hiding*, pp. 61–76, 2000. ISBN 978-3-540-46514-0.

Wu, P., Yang, Y., and Li, X. Stegnet: Mega image steganography capacity with deep convolutional network. *Future Internet*, 10:54, 06 2018. doi: 10.3390/fi10060054.

Ye, J., Ni, J., and Yi, Y. Deep learning hierarchical representations for image steganalysis. *IEEE Trans. on Information Forensics and Security*, 12(11):2545–2557, Nov 2017. ISSN 1556-6013. doi: 10.1109/TIFS.2017. 2710946.

Zhu, J., Kaplan, R., Johnson, J., and Fei-Fei, L. HiDDeN: Hiding data with deep networks. *CoRR*, abs/1807.09937, 2018.