SECURE IMAGE STEGANOGRAPHY USING GENERATIVE ADVERSARIAL NETWORKS

by

HIMA SAI KIRAN GUTTI	*411929*
RAHUL BANSAL	*411968*
VIKAS GARIKAPATI	*411924*

Under the guidance of

Dr. P. RAVEENDRA BABU

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY ANDHRA PRADESH
TADEPALLIGUDEM-534101, INDIA

MAY-2023

SECURE IMAGE STEGANOGRAPHY USING GENERATIVE ADVERSARIAL NETWORKS

Submitted in partial fulfillment of the requirements
for the award of degree
of

Bachelor of Technology

by

HIMA SAI KIRAN GUTTI	*411929*
RAHUL BANSAL	*411968*
VIKAS GARIKAPATI	*411924*

Under the Guidance of

Dr. P. RAVEENDRA BABU

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY ANDHRA PRADESH
TADEPALLIGUDEM-534101, INDIA

MAY-2023

PROJECT WORK APPROVAL

This project work entitled **"**SECURE IMAGE STEGANOGRAPHY USING GENERATIVE ADVERSARIAL NETWORKS**"** by HIMA SAI KIRAN GUTTI, bearing Roll No: 411929; RAHUL BANSAL, bearing Roll No: 411968 and VIKAS GARIKAPATI, bearing Roll No: 411924 is approved for the degree of Bachelor of Technology in the Department of Computer Science and Engineering.

Examiners

_____

_____

_____

Supervisor(s)

_____

Chairman

_____

Date:

Place:

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

HIMA SAI KIRAN GUTTI	RAHUL BANSAL	VIKAS GARIKAPATI
Roll No. 411929	Roll No. 411968	Roll No. 411924
Date:	Date:	Date:

CERTIFICATE

It is certified that the work contained in the thesis titled **"**SECURE IMAGE STEGANOGRAPHY USING GENERATIVE ADVERSARIAL NETWORKS**"** by HIMA SAI KIRAN GUTTI, bearing Roll No: 411929; RAHUL BANSAL, bearing Roll No: 411968 and VIKAS GARIKAPATI, bearing Roll No: 411924 has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Dr. P. RAVEENDRA BABU
Computer Science and Engineering
N.I.T. Andhra Pradesh
May, 2023

ACKNOWLEDGMENTS

We would like to express our gratitude to our supervisor, Dr. P. Raveendra Babu, who was a continual source of inspiration. He pushed us to think imaginatively and urged us to do this homework without hesitation. His vast knowledge, extensive experience, and professional competence in Cryptography enabled us to successfully accomplish this project. This endeavor would not have been possible without his help and supervision. We could not have asked for a finer supervisor in our studies. This initiative would not have been a success without the contributions of each and every individual. We were always there to cheer each other on, and that is what kept us together until the end. I am deeply grateful to everyone who has contributed to the successful completion of this project.

We would also like to express our gratitude to the faculty of the Department of Computer Science and Engineering, for providing such an opportunity for us and for their support.

HIMA SAI KIRAN GUTTI	RAHUL BANSAL	VIKAS GARIKAPATI
Roll No. 411929	Roll No. 411968	Roll No. 411924
Date:	Date:	Date:

ABSTRACT

Steganography is the art of hiding a hidden message inside of a regular message. Using a technique called image steganography, messages can be concealed within images. Steganography tries to conceal the existence of the message itself, unlike other approaches like cryptography that aim to stop attackers from reading the secret message. The Div2k dataset was used in this study to train a variety of models, and we present a novel method for concealing arbitrary binary data in images using generative adversarial networks. This method enables us to enhance the perceived quality of the images generated by our model.

TABLE OF CONTENTS

List of Figures

Figure **Page No.**

List of Tables

Table **PageNo.**

List of Abbreviations

CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
SSIM	Structural Similarity Index Measure
PSNR	Peak Signal to Noise Ratio
RS-BPP	Reed Solomon Bits-Per-Pixel

Chapter 1

Introduction

Steganography is the practice of concealing or hiding information within a carrier medium, such as an image, audio file, video, or text, in order to transmit a secret message without arousing suspicion.The term originated in the 15th century when messages were physically concealed. In modern times, steganography is used to secretly communicate digital messages. This involves placing a hidden message within a carrier, which can be a publicly visible medium. To increase security, the hidden message can also be encrypted, making it appear random and reducing the chances of it being discovered even if someone suspects its existence. If you want to learn more about steganography and the process of detecting hidden messages (steganalysis), there are several good introductory resources available in books and research papers [1-5].

Steganography has gained a lot of attention for its involvement in illicit activities, like planning and coordinating crimes through hidden messages in publicly posted images. This makes it difficult for authorities to discover both the communication and its intended recipient [6]. However, beyond these negative uses, there is a legitimate use case for steganography where it is employed to embed authorship information, such as digital watermarks, without compromising the integrity of the content or image.

The most common techniques used in steganography involve manipulating the least significant bits (LSB) of images to hide secret information. This can be done in different ways, either uniformly or adaptively, using simple replacement or more sophisticated methods [9, 10]. Although these alterations are often not visually detectable, analyzing the statistics of image and audio files can reveal any deviations from their original form. Advanced approaches aim to maintain the statistical characteristics of the cover images by creating and matching models based on first and second-order statistics. One well-known method in this regard is called HUGO [11], which is commonly used for hiding relatively small messages (less than 0.5 bits per pixel). In contrast, our study takes a different approach by employing a neural network to implicitly learn the distribution of natural images. Moreover, we go beyond

small messages and explore the embedding of much larger messages, such as full-size images, into the carrier image.

Despite recent impressive results achieved by incorporating deep neural networks with steganalysis [12–14], there have been relatively few attempts to incorporate neural networks into the hiding process itself [15–19]. Some of these studies have used deep neural networks (DNNs) to select which LSBs to replace in an image with the binary representation of a text message. Others have used DNNs to determine which bits to extract from the container images.The neural network determines where to place the secret information and how to encode it efficiently; the hidden message is dispersed throughout the bits in the image. A decoder network, that has been simultaneously trained with the encoder, is used to reveal the secret text. Note that the networks are trained only once and are independent of the cover image and secret text.

We have trained different models using Div2k dataset and conducted an analysis on results to find out the best model that is suitable for hiding text in an image.The steganographic models we used in our project are LSB steganographic Model,SteganoGAN model, Pixel Value Difference(PVD) model and Autoencoder-Decoder model.

Chapter 2

Literature Review

In order to totally conceal the fact that a message is being transmitted, steganography is the technique of enclosing messages or information in messages or images that appear harmless. Steganography has advanced in sophistication along with the development of digital technology. Today, data is hidden within photos, audio, video, and other forms of digital media using steganographic techniques.The concealed information is intentionally concealed so that it cannot be seen by untrained eyes.

Data is often concealed using traditional steganography techniques within digital material, such photos, music, or video. Least Significant Bit (LSB) replacement is a well-liked method for encoding secret messages in images. This method modifies the least significant bits of the pixel values in an image.

An active area of research in recent years has been the use of neural networks in steganography. New steganography methods that are more effective and secure than conventional ones have been created using neural networks. We'll talk about numerous steganography methods in this section.

Because embedding a message can change the carrier's look and underlying statistics, good steganography is a difficulty. Two things determine how much of an alteration there will be: first, how much information will be hidden. Text messages have frequently been concealed in graphics. In bits-per-pixel (bpp), the quantity of information that is concealed is expressed. The information level is frequently adjusted to 0.4bpp or less. The larger the bpp and hence the more the carrier is altered, the longer the message is [6, 7]. Second, the carrier picture itself determines how much is altered. Less human observable perturbations result from information hiding in noisy, high-frequency filled portions of an image as opposed to flat regions. Work on estimating how much information a carrier image can hide can be found in [8].

2.1 LSB Steganography

Least Significant Bit (LSB) steganography is one of the simplest and most widely used techniques for hiding information within digital media, such as images, audio, and video. The basic idea behind LSB steganography is to modify the least significant bits of the pixel values in an image or the samples in an audio file to encode the hidden message.

In LSB steganography, the binary representation of each message character is embedded into the binary representation of a pixel or a sample in the digital media. The embedding process involves replacing the least significant bits of the pixel or the sample with the bits of the message character. Since the least significant bits of a pixel or a sample contribute the least to the overall value, they can be modified without significantly changing the appearance or the quality of the media.

For example, consider an image with a pixel value of (210, 178, 132) in RGB format. The binary representation of each component is as follows:

210 = 11010010
178 = 10110010
132 = 10000100

Now, suppose we want to hide the message "HELLO" in this image. The binary representation of "HELLO" is:

H = 01001000
E = 01000101
L = 01001100
L = 01001100
O = 01001111

To embed the message, we start with the first pixel value (210, 178, 132), and replace the least significant bit of each component with the bits of the message character in order. The resulting pixel value becomes (211, 178, 133). We then move to the next pixel value and repeat the process until we have embedded the entire message.

To extract the hidden message from the stego-image, we simply extract the least significant bits of each pixel value and concatenate them to form the message characters.

While LSB steganography is easy to implement and can hide information in plain sight, it is not very secure and can be easily detected and decoded by steganalysis techniques. Therefore, more sophisticated steganography techniques, such as those based on neural networks or transform domain methods, are often used for more secure and robust information hiding.

2.2 Steganography using Pixel Value Difference(PVD)

A steganographic method called Pixel Value Differencing (PVD) is used to conceal sensitive data in digital photographs. In order to encode secret data without drastically modifying the image's appearance, it is necessary to modify the least significant bits (LSBs) of the pixel values in the image. PVD compares and calculates the difference between two successive pixel values in a picture. The LSB of the current pixel is left unaltered if the difference is even, and it is flipped to match the parity of the difference if the difference is odd. The secret data is embedded in the LSBs of the pixel values after repeating this method for each and every pixel in the image.

It is a technique for hiding messages in images that makes use of the difference between horizontal and vertical pixels. In the horizontal direction, we choose two pairs of horizontal pixels to hide messages using a high-quality model function method, and in the vertical direction, we choose one pair of vertical pixels to hide messages using conventional PVD methods.

In the PVD scheme, the cover image is divided into non overlapping blocks, where each block consists of two consecutive pixels. The data hiding procedure is independent in each block. Denote the pixel pair in some block as P_i and P_j the difference d is calculated as $d = P_j - P_i$ with |d| ϵ [0, 255]. This range is divided into r subregions $R_k(k = 1, 2 \ldots r)$, the width of each sub-region is power of 2. The number of bits embedding in the region of R_k is $n = log_2(w_k)$, where w_k is the width of the k-th subregion.

2.3 Steganography using Deep Neural Network(DNNs)

Steganography using deep neural networks is an emerging area of research that aims to develop more secure and robust steganography techniques. DNNs are capable of learning complex relationships between features in an image, and can be used to create more sophisticated and secure steganographic techniques.

One approach that has been used in steganography is the use of autoencoders, which are neural networks that learn to encode and decode data. In steganography, an autoencoder can be trained to embed a secret message in an image by encoding the message and then combining it with the original image. The resulting stego-image can be decoded using the same autoencoder to recover the hidden message.

Another approach is to use DNNs to learn the statistical properties of an image and hide information in specific regions of the image that are less likely to be detected. For example, a DNN can be trained to identify regions of an image that are perceptually important and therefore more likely to be inspected by steganalysis techniques. The hidden message can then be embedded in the less perceptually important regions of the image, making it less likely to be detected.

Overall, the use of DNNs in steganography has shown promising results and has the potential to improve the security and efficiency of steganography techniques. However, there is still ongoing research in this field, and the use of DNNs in steganography is an active area of research. In the paper *Hiding Images in Plain Sight: Deep Steganography* [20], the deep neural network built using the ImageNet dataset determines where to place the secret information and how to encode it efficiently; the hidden message is dispersed throughout the bits in the image. A decoder network, that has been simultaneously trained with the encoder, is used to reveal the secret image. Note that the networks are trained only once and are independent of the cover and secret images.

The network architecture has three components :
(1) Prep network: To extract the features from the secret image provided as input.

(2) Hidden Network: Encodes the output of Prep Network into the cover image to produce the container image.

(3) Reveal Network: Decodes the secret image from the container image.

The system is trained by reducing the error shown below (c and s are the cover and secret images respectively, and β is how to weigh their reconstruction errors):

$$L(c, c_0, s, s_0) = ||c - c_0|| + \beta||s - s_0|| \quad (2.1)$$

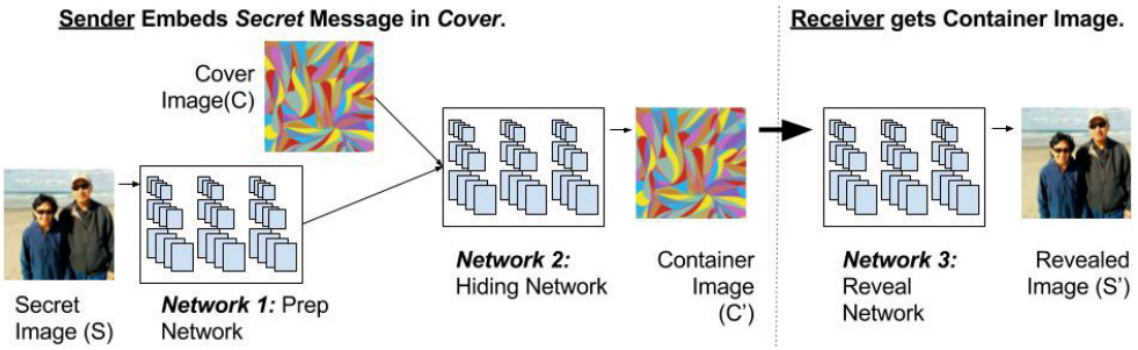

Figure 2.1 Model Architecture using DNNs[20]

The objective of this work is to visually conceal an entire NxNxRGB secret image in another NxNxRGB cover image with the least amount of distortion possible (each colour channel is 8 bits). We remove the criterion that the secret image is received losslessly, in contrast to earlier investigations, where a hidden text message must be sent with complete reconstruction.

Instead, we are open to making compromises regarding the carrier's and covert image's quality. We also offer succinct discussions about the likelihood that the existence of the hidden message will be discovered. Previous research has shown that it is possible to find buried message bit rates as low as 0.1bpp; our bit rates are 10–40 times greater. We do not anticipate the existence of a secret message to be hidden from statistical analysis, despite being visually difficult to discern, given the substantial amount of hidden information. We will still demonstrate that widely used methods are unable to locate it and provide encouraging suggestions for how to balance the difficulty of existence discovery with reconstruction quality if necessary.

The steganography images created using this method cannot be recognised by the human eye or by any common steganalysis software. In contrast, when we train a steganalysis network as binary classifiers, we use our containers as positive examples and the unaltered ImageNet images as negative samples. The cumulative classification rates for the fully convolutional networks ranged from 90 to 95%.For contrast, the same networks were retrained to identify least-significant-bit substitution, which involves swapping out the L least significant bits from each colour channel in the cover image for the L most significant bits from the hidden image. The networks had cumulative success rates of over 99% when they were trained and tested with L = 1, 2, 3, and 4. Additional tests showed that the detection rate remained above 99% even when a single bit was arbitrarily assigned to one of the 4 bit places. The high detection rates with a whole image are not unexpected given these detection rates on even a single bit.

In this study, the steganalysis networks were trained and used after the complete encoding system was created. However, using pre-trained and/or simultaneously trained steganalysis networks in an adversarial learning framework, such as Generative-Adversarial-Networks (GAN), during the training of steganography nets provides a method to incorporate an obfuscation based error metric. The adversary provides a supplemental error signal based upon the discoverability of the hidden message that is minimized in addition to the reconstruction errors.

2.4 Steganography using Generative Adversarial Networks

Steganography using Generative Adversarial Networks is a relatively new and exciting area of research that has shown promising results in hiding information within digital media, particularly images. GANs are a type of deep neural network that can generate realistic images by learning the underlying distribution of a dataset.

The basic idea behind steganography using GANs is to train a GAN to generate images that contain a hidden message. This is achieved by modifying the training process of the GAN to include a loss term that encourages the generator to produce images that embed the hidden message.

One approach that has been used in steganography using GANs is to modify the generator network to include a secret message embedding module. This module takes as input both the

generator's output and the message to be embedded and modifies the output of the generator to embed the message in a way that is imperceptible to the human eye.

Another approach is to modify the discriminator network to detect the presence of a hidden message in the generated images. This is achieved by training the discriminator to distinguish between images that contain a hidden message and those that do not. The generator is then trained to produce images that fool the discriminator into thinking that they do not contain a hidden message.

One advantage of using GANs in steganography is that the generated images are usually more realistic and visually pleasing than those generated by other steganography techniques. Additionally, GANs can be trained to generate images with specific attributes, such as texture or color, which can be used to hide the hidden message in a more secure and robust way.

However, steganography using GANs is still an active area of research, and there are several challenges that need to be addressed. One challenge is to ensure that the hidden message is imperceptible to the human eye while still being detectable by steganalysis techniques. Another challenge is to ensure that the generator produces images that are both visually pleasing and contain the hidden message.

Chapter 3

Proposed Approach

In order to hide a message inside an image a Generative Adversarial Network has been used. The implementation maintains the architecture of the networks originally proposed by the authors of the paper titled *SteganoGAN:High Capacity Image Steganography with GANs*. The overall architecture is composed of three networks namely, an encoder, a decoder and a critic. The encoder takes an image, also called as cover throughout the report, and hides a message inside it; the decoder takes the image generated by the encoder and extracts out the hidden message; the critic provides a quantitative measure of the quality of the cover and the steganographic image. A detailed explanation of the three networks has been provided below:

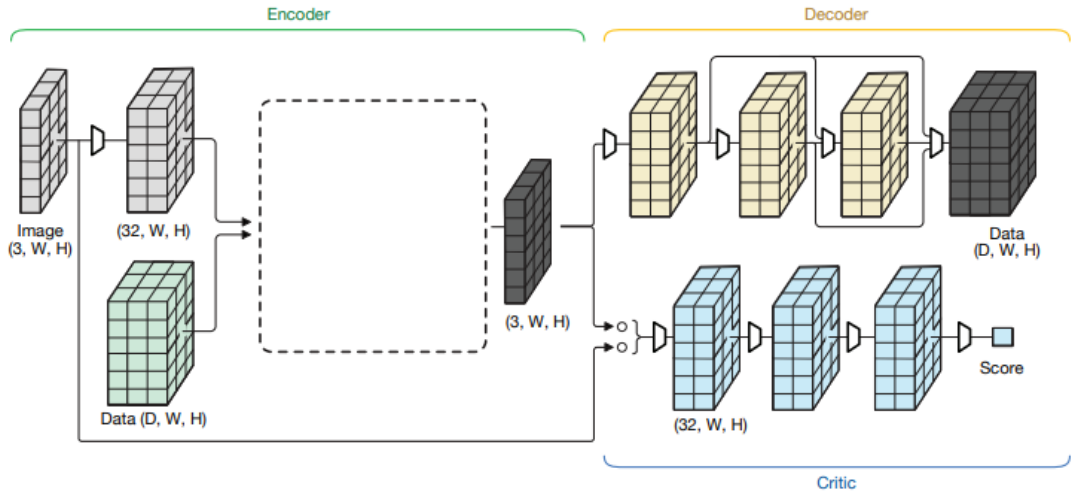

Figure 3.1: Overall architecture of the model including encoder, decoder and critic[5]

3.1 Encoder

The encoder network's responsibility is to take a cover image (C) and enter a text that the user wants to conceal. The text is converted into a binary data tensor $M \in \{0, 1\}^{DxWxH}$. Here, D stands for the number of bits that we want to conceal in each pixel of the cover image, and W&H stand for the cover image's width and height.

We experimented with three variants of the encoder architecture. The three encoders differ in the connectivity patterns. The following two operations are applied to both the encoders at the start:

1. The cover image is converted to a tensor a using convolutional block:

$$a = Conv3{\rightarrow}32(C) \tag{3.1}$$

2. The message M is concatenated to a and the result is processed with a convolutional block. The resulting tensor is called b:

$$b = Conv32 + D{\rightarrow}32(Cat(a, M)) \tag{3.2}$$

3.1.1 Basic:

In the basic encoder the tensor b generated above is treated with convolutional blocks. The output from these operations is the steganographic image. We represent it as follows:

$$E_b(C, M) = Conv32{\rightarrow}3(Conv32{\rightarrow}32(b)) \tag{3.3}$$

3.1.2 Residual:

The basic encoder discussed above can be modified by adding the cover image C to its output. This helps the encoder to learn to produce a residual image as shown in Figure:3.2(middle). This is hypothesized to improve the quality of the steganographic image. The above operation is represented as follows:

$$E_r(C, M) = C + E_b(C, M) \tag{3.4}$$

3.1.3 Dense:

In this type of encoder additional connections between the convolutional blocks are introduced. This makes it possible to concatenate the feature maps generated by the earlier blocks to the maps generated by the later blocks. This is done to improve the embedding rate. Below we represent the same formally:

$$c = Conv64 + D{\rightarrow}32(Cat(a, b, M))$$
$$d = Conv96 + D{\rightarrow}3(Cat(a, b, c, M)) \quad (3.5)$$

$$E_d(C, M) = C + d$$

Thus the output of each variant is a steganographic image $S = E\{b, r, d\}(C, M)$. This has the same resolution and depth as that of the cover image C.

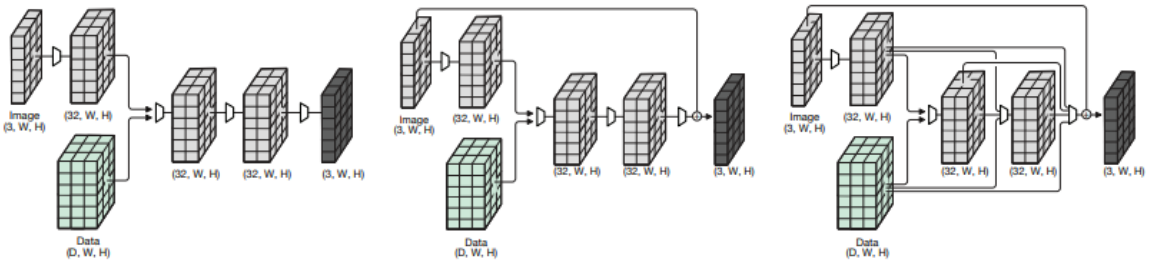

Figure 3.2: The three variants of Encoder, Basic (left); Residual (middle); Dense (right)[5]

3.2 Decoder

The input to the decoder is the steganographic image generated by the encoder. It tries to recover the data tensor M which is the original and, therefore, produces a data tensor $\widehat{M}$. The decoder can be expressed as:

$$a = Conv3{\rightarrow}32(S)$$
$$b = Conv32{\rightarrow}32(a)$$
$$c = Conv64{\rightarrow}32(Cat(a, b)) \quad (3.6)$$
$$D(S) = Conv96{\rightarrow}D(Cat(a, b, c))$$

3.3 Critic

The critic network has three convolutional blocks followed by a convolutional layer with one output channel. In order to produce a scalar score an adaptive mean pooling operation is applied to the output of the convolutional layer. The operations are as follows:

$$a = Conv32\rightarrow32(Conv32\rightarrow32(Conv3\rightarrow32(S))) \quad (3.7)$$
$$C(S) = Mean(Conv32\rightarrow1(a))$$

This network is used to provide feedback on the performance of the encoder and ultimately generate more realistic images.

3.4 Evaluation metrics

Steganography algorithms are evaluated along three aspects:

- *capacity:* the amount of data that can be hidden in an image,
- *distortion:* the similarity between the cover and steganography image,
- *secrecy:* the ability to avoid detection by steganalysis tools.

In this section, we are going to describe some metrics for evaluating the performances of the models along these aspects.

3.4.1 Reed Solomon bits-per-pixel (RS-BPP):

First, we want to determine the actual amount of bits that may be transmitted per pixel in order to ensure that the method is effective in terms of capacity. The number may seem to be proportionate to the number of pixels, but recovering a hidden bit depends significantly on the model, the cover image, as well as the message itself, thus the issue is more complicated than it first appears.

This means that the unique information of the probability p that our decoder can mislead a single bit is meaningful, we may think that the payload length would be simply multiplying $1 - p$ times the length of the message, but it is not so trivial, thus, we have to introduce a mechanism to recover the information hidden into the pixels. For this reason we take advantage of the Reed Solomon Code, which is a cyclic non binary code for the detection and reconstruction of the error. This algorithm, based on the assumption that for k distinct points it's possible to reconstruct a polynomial of $k - 1$ degree, produces an information of length k a second information of length m for the recovery of the error. Thus the total payload is $n = k + m$. Now we know that our encoder will generate an information of total size n, in which is embedded the message we want to transmit of size k ($n \geq k$). Reed Solomon Code guarantees that $(n - k)/2 = m/2$ errors can be recovered. This implies that given a

steganographic algorithm that produces an incorrect bit with probability p, we want the number of incorrect bits to be less or equal the number of bits we can correct:

$$p.n \leq (n-k)/2 \Rightarrow k/n \leq 1-2p \tag{3.8}$$

Where k/n is the average bit of the real message we want to transmit with respect to the total message length. In this way we can multiply this ratio times the number of bits we want to hide in each pixel to recover the "real" payload length we are able to transmit and recover.

We refer to this metric as Reed Solomon bits-per-pixel (RSBPP), 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.

3.4.2 Peak to Signal Noise Ratio (PSNR):

In order to evaluate the distortion introduced by the steganographic algorithm it's useful to introduce another measure called 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.

Given two images X and Y of size (W, H) and a scaling factor γ which represents the maximum possible difference in the numerical representation of each pixel (for example, if the images are represented as floating point numbers in $[-1.0, 1.0]$, then $\gamma = 2.0$ since the maximum difference between two pixels is achieved when one is 1.0 and the other is -1.0).

The PSNR is defined as a function of the mean squared error (MSE):

$$\text{MSE} = \frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H}(X_{i,j} - Y_{i,j})^2 \tag{3.9}$$

$$\text{PSNR} = 20.log_{10}(\gamma) - 10.log_{10}(MSE) \tag{3.10}$$

3.4.3 Structure Similarity Index Measure (SSIM):

Structural similarity index is a second similarity metric between the cover image and the steganographic one. Given two images X and Y, the SSIM can be computed using the means μX and μY, with variance σ_X^2 and σ_Y^2 and the covariance σ_{XY}^2

$$SSIM = \frac{\left(2\mu_X\mu_Y + k_1^2R\right)\left(2\sigma_{XY}^2 + k_2^2R\right)}{\left(\mu_x^2 + \mu_Y^2 + k_1^2R\right)\left(\sigma_X^2 + \sigma_Y^2 + k_2^2R\right)} \quad (3.11)$$

The default configuration for SSIM uses $k1 = 0.01$and $k2 = 0.03$ and returns values in the range $[-1.0, 1.0]$ where 1.0 indicates the images are identical.

Chapter 4

Experimental Procedure

4.1 Platform used

The task at hand required us to train GAN models for experimentation and analysis. Thanks to Google Colab we could use the Google servers and GPU to efficiently train our models.

4.2 Dataset description

We trained all our models on the Div2k image dataset by Agustsson & Timofte. The dataset is well structured and contains all sorts of landscape as well as portrait images. We used the default train/test split proposed by the creators of the Div2k. The train dataset contains 800 images with a total size of 3.5GB. On the other hand the validation set contains 200 images with a total size of 450MB.

4.3 Models Experimentation

We trained the following 4 models on the div2k dataset to check the most suitable model for hiding arbitrary data into the image. We shall discuss these models, how they are trained, in the following subsections.

4.3.1 LSB Steganographic Model

It consists of three parts: hiding text in an image, revealing hidden text from an image, and evaluating the quality of the steganographic process using metrics such as MSE (Mean Squared Error), PSNR (Peak Signal-to-Noise Ratio), and SSIM (Structural Similarity Index).

- Hiding Text in an Image: The hide_text_in_image function takes a text message, an image_path for the input image, and an output_path to save the modified image. It

uses LSB steganography to hide the binary representation of the text message within the RGB pixels of the image. The least significant bits (LSBs) of each color channel are modified to match the corresponding bits of the text message. The modified image is then saved to the specified output_path.

- Revealing Hidden Text from an Image: The reveal_text_in_image function takes an image_path as input and aims to extract the hidden text from the steganographically modified image. It iterates over the pixels of the image, retrieving the LSBs of the RGB values and concatenating them to form the binary representation of the hidden text. Finally, it converts the binary text into ASCII characters and returns the extracted message.

4.3.2 PVD Model

The embedding process is a key step in image steganography, which involves hiding secret data within an image. In this process, a reference image and a secret file are read as input. The reference image is then divided into non-overlapping blocks of [3x3] pixels.

For each block, the central pixel is selected as a reference pixel, and all other pixels in the block (excluding the reference pixel) are iterated through. For each non-reference pixel, the pixel difference between it and the reference pixel is computed in each color channel (R, G, and B).

The pixel differences are then used to determine the number of Least Significant Bits (LSBs) in each channel of the non-reference pixel that can be replaced with the bits from the secret file. The required number of bits from the secret file is read for each channel, and the corresponding LSBs of the non-reference pixel are replaced with the bits from the secret file.

This process is repeated until all the bits from the secret file are embedded in the reference image. Once the embedding process is complete, the modified reference image is saved as the output image.

The embedding process can be used for a variety of applications, including information hiding, watermarking, and copyright protection. It is important to use this technique ethically and responsibly, taking into account the potential risks and limitations. It is also essential to

ensure that the steganographic image is not detectable by unauthorized parties, as this could compromise the security of the hidden data.

We are loading the reference image and the PVD image. Check if the dimensions of the two images match. if the differences don't match we will throw an error. Divide the reference image into non-overlapping 3x3 matrices. For each matrix, calculate the difference between the center pixel and the other pixels in the matrix.

Use the PVD table to determine how many bits are required to encode the difference. If the difference between pixels is less than 16 we can change 2 bits from LSB. If the pixel difference is between 16 and 32 we can change 3 bits, else if the pixel difference is greater than 32 we can change 4 bits from LSB.

Then we will Extract the least significant bits from the corresponding pixel in the PVD image. Write the extracted bits to a file.

If the header of the PVD file has not been read yet, read the header to obtain the PVD version and the size of the encoded data. Keep writing the extracted bits until the size of the encoded data is reached. and Close the output file and return the total number of bits extracted.

4.3.3 Autoencoder-Decoder Model

We have trained an autoencoder-decoder model with the Div2k dataset. The network architecture has two components:

- Encoder: The Encoder network takes the data and the cover image as input and produces the steganographic image as output. The encoder model is inspired from the Dense Net architecture. The Encoder in the code also utilizes dense connections by concatenating the output of previous layers with the input of the current layer. This allows for the network to learn richer representations by using the output of previous layers in the current layer's computation.

- Decoder: The aim of the decoder is to decode an embedded data tensor from a given steganographic image. We have two convolutional layers, and a new third convolutional layer is added that takes the concatenation of the output of the first and second convolutional layers as its input. Then, a final convolutional layer takes the concatenation of the outputs of all three convolutional layers as its input to output the decoded data tensor.

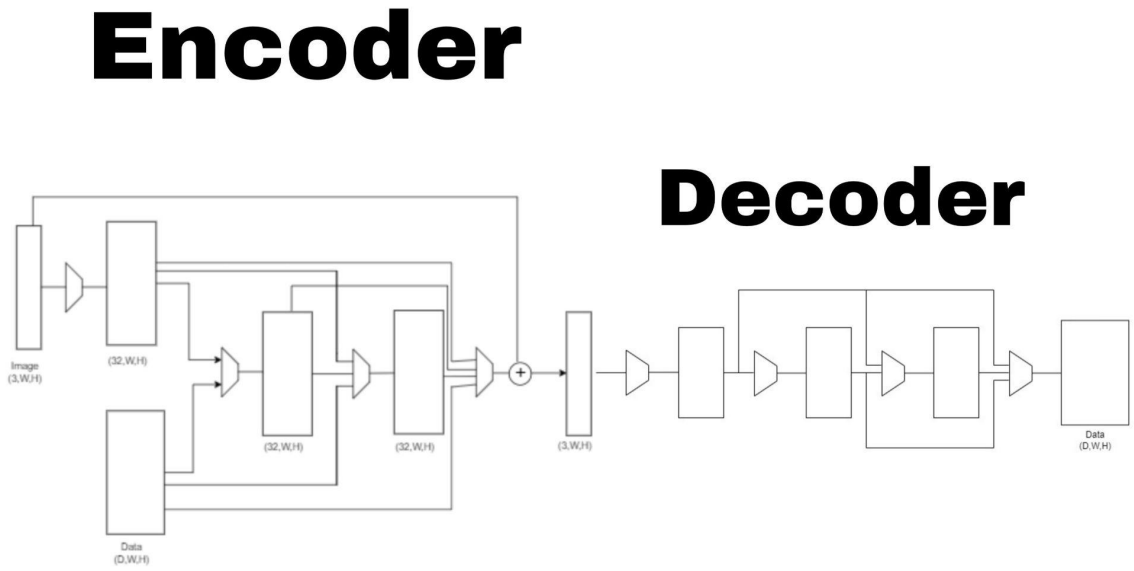

Figure 4.2.: Network architecture of Autoencoder decoder network

We have trained the autoencoder decoder model for 32 epochs using ADAM optimizer with a batch size of 4, learning rate = 0.0001 and data depth = 4. In order to generate good results we need to

optimize the autoencoder decoder network. We perform the optimization iteratively. The optimization of the encoder-decoder network is done by optimizing two losses simultaneously. These are:

- decoder loss:

$$L_d = E_{X \sim p_c} CrossEntropy(D(\varepsilon(X, M)), M) \quad (4.1)$$

- mean square error:

$$L_s = E_{X \sim P_c} \frac{1}{3 \times W \times H} ||X - \varepsilon(X, M)||_2^2 \quad (4.2)$$

4.3.4 SteganoGAN Model

The network architecture is similar to that of the previous autoencoder-decoder model except, we have added a critic network to it. The critic network provides feedback for the encoder model which can moduled into an extra component of the loss function.

Generative Adversarial Networks are capable of producing extremely good results, but training a GAN is quite challenging. The fact that all the sub-networks involved, Generator and Discriminator, are trained simultaneously and training of one affects the other. It is often the case that improvements to one sub-network come at the expense of the other sub-network.

Generally it is recommended to not opt for a large batch size for training a GAN network. The reason being that in the initial phases of training the discriminator gets a lot of examples to train on because of a larger batch size which can overpower the generator. This can lead to an unstable network and the network as a whole might not converge. Hence, we have trained our model on a batch size of 4.

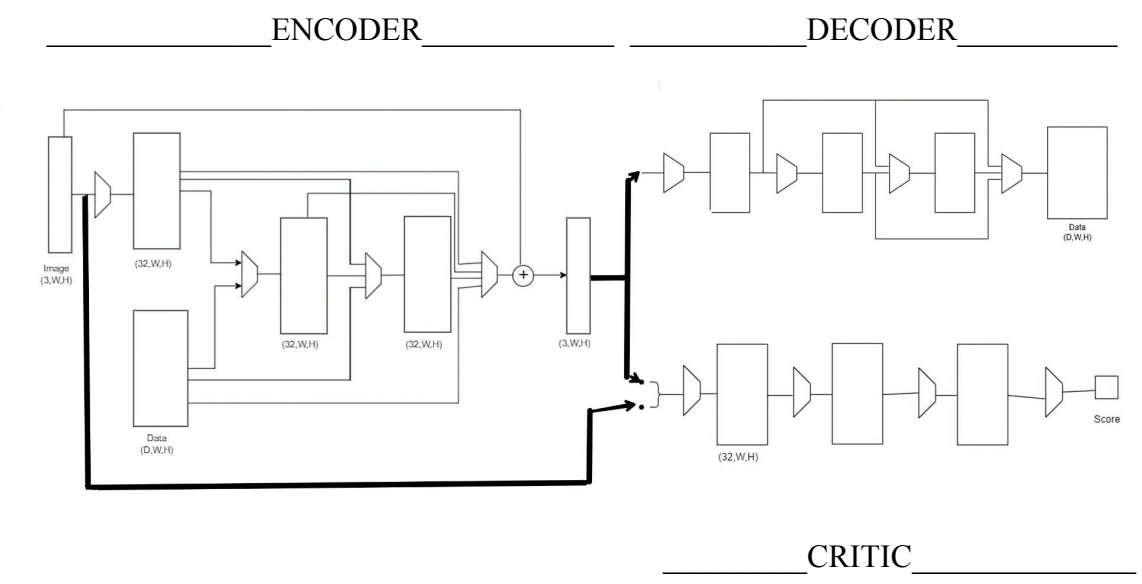

Figure 4.3.: Network architecture of SteganoGAN model

We have trained the model using a random bit vector as payload and images from the Div2k dataset as the cover images. We have trained the model at different data depths(data depth is the hyperparameter which determines no. of bits we want to store for each pixel). We have trained the model for 32 epochs using ADAM optimizer at a learning rate = 0.0001. Among

all models trained at different data depths, the model trained at data depth = 4 seemed to be more effective. Our goal is to hide a given text into the given cover image. Since the model is trained on a random bit vector of the size $D * W * H$, we have to generate a bit vector of size $D * W * H$ from the text given to us. And also, for error free transmission, we have Reed Solomon Encoding. At first, the text will be converted to a byte array, this byte array is passed to the reed solomon encoder which adds some parity bytes. The resultant byte array is converted to a bit vector. The bit vector is repeatedly concatenated with itself with a separation of vectors of zeros between each bit vector. This process is done until we get a vector of size $D * W * H$ which will be the required payload. The payload concatenated with the input generated from the image will be provided as input to the encoder which will generate the steganographic image.

The decoder network takes the steganographic image and decodes the payload. The payload is split based on the vector of zeros used while concatenating the bit vectors. After splitting, we will be getting a list of bit vectors. These bit vectors are decoded using Reed Solomon decoder which corrects any errors that have occurred while transmission and a list of text values will be generated. Out of this list of text values, the text which has occurred more frequently will be the decoded text message.

In order to generate good results we need to optimize the three networks involved namely, encoder, decoder and critic. We need to perform optimization for the two loss functions equation 4.1, 4.2 and also for another loss function for the critic network simultaneously

- realness score of steganographic image:

$$L_r = E_{X \sim P_c} C(\varepsilon(X, M)) \tag{4.3}$$

Now, while training the objective function:

$$minimise\ L_d + L_s + L_r \tag{4.4}$$

The critic network is trained by minimizing the difference between the score of the cover image and the score of the corresponding steganographic image. The score comes from the critic network.

$$L_c = E_{X \sim p_c} C(X) - E_{X \sim P_c} C(\varepsilon(X, M)) \tag{4.5}$$

Chapter 5

Results and discussion

In the section, we shall discuss various results achieved while conducting the experiments on the models.

5.1 Evaluation of various models

We have tested different models using the same Div2k dataset and with the same text message saying “Code Red, Abort!!!”. We have compared the outputs generated by these models with the same set of evaluation metrics. In the following four subsections, we shall discuss the metric values we have obtained.

5.1.1 LSB Model results

When the text message is embedded into a cover image using LSB steganography we have obtained the following results:

PSNR: 100.789

SSIM: 0.999

BPP: 1.000

Decoder Accuracy: 100.000

We have achieved high PSNR and SSIM values. When we try to increase the text length or the no. of least significant bits to be replaced per each pixel, the image is getting more distorted. LSB steganography has good SSIM and PSNR metric values but it is impotent to the powerful steganalysis tools like StegExpose. The hidden text message can be easily retrieved using the steganalysis tools. And also, if we have trained a binary classifier with the original cover images as positive samples and the corresponding steganographic images as negative samples. Given an image, it can be easily classified with a detection rate of over 99%.

5.1.2 PVD Model results

When the text message is embedded into a cover image using PVD steganography we have obtained the following results:

PSNR: 84.552

SSIM: 0.999

BPP: *depends on cover image and PVD implementation*

Decoder Accuracy: 100.000

Let the histogram of d, which is the difference between two pixels in a block, be h(d) $-255 \leq d \leq 255$. Generally speaking, in a normal image, the number of occurrences of the pixel difference, h(d), decreases with increasing |d| in a macroscopically smooth fashion. Whereas the histogram of a steganographic image generated using the PVD approach is not smooth.

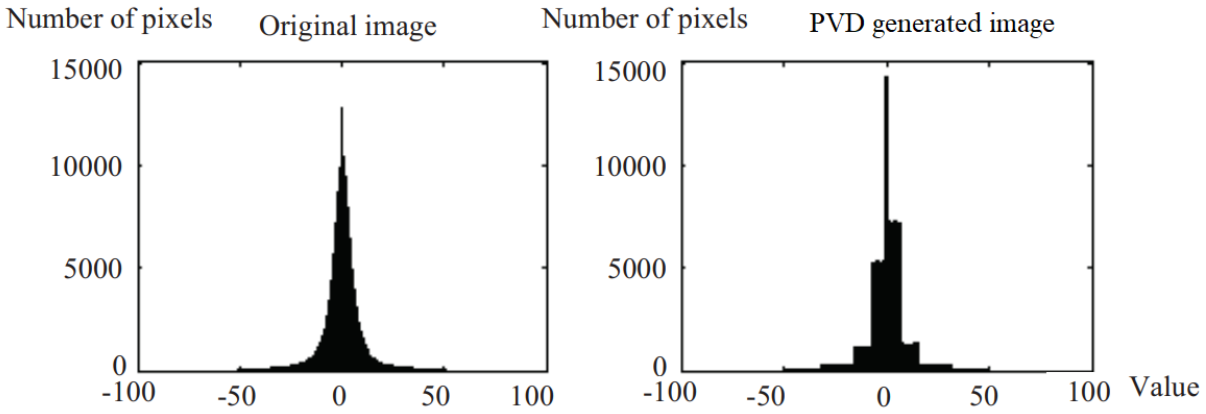

Figure 5.1: Pixel difference histogram of stego-image and original image[25].

5.1.3 Autoencoder decoder Model results

We have trained the autoencoder decoder model using the Div2k training set at a data depth of 4 for 32 epochs using ADAM optimizer at a learning rate = 0.0001 using a batch size of 4 and obtained the following results:

PSNR: 22.948

SSIM: 0.309

BPP: 3.966

Decoder Accuracy = 0.996

We have achieved a good BPP value and decoder accuracy, but the model has poor SSIM and PSNR metric values. The autoencoder-generated steganographic images are more potent against detection by StegExpose than simpler steganography methods, it is not a guarantee that they will always evade detection. When we take a binary classifier trained on original images as positive samples and the steganographic images as negative samples. Given an image, the classifier can be classified with a detection rate of over 90-95%.

5.1.4 SteganoGAN Model results

We have trained the steganoGAN model using the Div2k training set at a data depth of 4 for 32 epochs using ADAM optimizer at a learning rate = 0.0001 using a batch size of 4 and obtained the following results:

PSNR: 37.429

SSIM: 0.890

BPP: 2.285

Decoder Accuracy = 0.786

We have achieved quite good values for PSNR and SSIM when compared to the autoencoder decoder model. The BPP value (= 2.285) has reduced. However, it is quite manageable since we are using reed solomon correction codes to correct the bits that are wrongly transmitted. The detection of the steganographic images produced using steganoGANs by the StegExpose tool is difficult when compared to all the previous models. Since we are training a critic network along with the autoencoder decoder network, which provides feedback on the work of the encoder, the detection rates for a binary classifier are quite low. Using the SteganoGAN model we achieved better results compared to other standard models.

5.1.4.1 Comparison of SteganoGAN model at different data-depths

For a simple comparison we select four models out of all the trained sets of models. Each of the four selected models were trained for 32 epochs, but at different data depths namely, 2, 4, 6 & 8. Each of the selected models is provided with a sequence of images which were not

shown to the model during the training phase. The results from experiments conducted for the analysis of the selected models trained are shown in Table : 5.1.

Performance on Dense Encoder and Dense Decoder								
D	RS-BPP		PSNR		SSIM		Decoder Accuracy	
	OD	ND	OD	ND	OD	ND	OD	ND
2	1.96	1.90	39.62	38.01	0.92	0.89	-	0.98
4	2.53	2.29	37.49	37.43	0.88	0.89	-	0.79
6	2.44	2.04	38.94	38.71	0.90	0.93	-	0.67
8	-	2.19	-	39.69	-	0.93	-	0.63

Table 5.1: Performance of various SteganoGAN models on validation dataset each trained for 32 epochs at different data depths (OD: Original Data reported in paper;ND: New Data obtained; -: Original data not reported in paper).

From the table:5.1 it can be seen that a model trained for data depth = 4 achieves a greatest RS-BPP. This means that the effective number of bits that can be conveyed per pixel is highest for the model trained for d = 4. At the same time the model has a respectable SSIM as well as the accuracy of the decoder to extract out the hidden message from the steganographic image is quite high.

5.1.4.2 Hiding a text in the image using SteganoGAN model

We tried to encode a text message into an image taken from the training dataset as a cover image. We first converted the text message into a bit-vector of fixed length and fed it to the network. The bit vector is encoded into the image by the encoder network which can be

decoded at the decoder end; We also used a critic network to discriminate between the cover image and the image obtained after encoding the textual message ino the image. The following is the result obtained when we tried to encode a text into cover image:

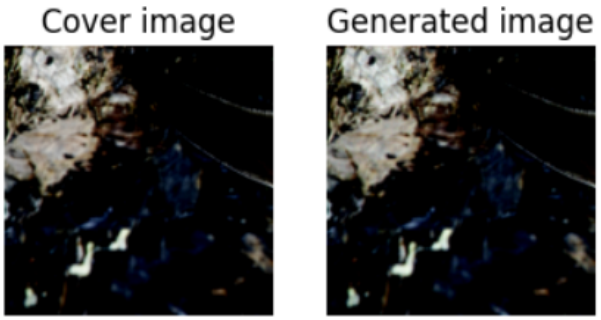

Figure 5.2: The image is encoded with a text : “Style too own civil out along”. The image on the left is the original and the image on the right is the image obtained after encoding the information.

Chapter 6

Summary and Conclusions

In this project, we have seen in detail various evaluation metrics used to evaluate the performance of the steganography models. We have compared the outcomes of various text-image steganography models using the same set of evaluation metrics. We have found that the steganoGAN model is found to yield better results when compared with the other steganographic models discussed. We tried to achieve high payloads while evading detection to bare eye and steganalysis tools using the steganoGAN model. As part of the future work, we can further investigate the generative adversarial approaches to steganography exploring new architectures that can come up with some better results and higher payloads.

References

[1] Kessler, G. C. & Hosmer, C. An overview of steganography. Advances in Computers, 83, 51-107,(2011).

[2] Kessler, G. C. An overview of steganography for the computer forensics examiner. Forensic science communications, 6(3), 1-27,(2004).

[3] Jussi Parikka. Hidden in plain sight:The steganographic image. https://unthinking.photography/themes/fauxtography/hidden-in-plain-sight-the-steganographic-image, 2017.

[4] Jessica Fridrich, Jan Kodovsk`y, Vojtˇech Holub, and Miroslav Goljan. Breaking hugo–the process discovery. In International Workshop on Information Hiding, pages 85–101. Springer, 2011.

[5] Zhang, Kevin Alex et al. “SteganoGAN: High Capacity Image Steganography with GANs.” ArXiv abs/1901.03892 : n. pag,(2019).

[6] Jessica Fridrich and Miroslav Goljan. Practical steganalysis of digital images: State of the art. In Electronic Imaging 2002, pages 1–13. International Society for Optics and Photonics, 2002.

[7] Hamza Ozer, Ismail Avcibas, Bulent Sankur, and Nasir D Memon. Steganalysis of audio based on audio quality metrics. In Electronic Imaging 2003, pages 55–66. International Society for Optics and Photonics, 2003.

[8] Farzin Yaghmaee and Mansour Jamzad. Estimating watermarking capacity in gray scale images based on image complexity. EURASIP Journal on Advances in Signal Processing, 2010(1):851920, 2010.

[9] Jessica Fridrich, Miroslav Goljan, and Rui Du. Detecting LSB steganography in color, and gray-scale images. IEEE multimedia, 8(4):22–28, 2001.

[10] Abdelfatah A Tamimi, Ayman M Abdalla, and Omaima Al-Allaf. Hiding an image inside another image using variable-rate steganography. International Journal of Advanced Computer Science and Applications (IJACSA), 4(10), 2013.

References

[11] Tomáš Pevn`y, Tomáš Filler, and Patrick Bas. Using high-dimensional image models to perform highly undetectable steganography. In International Workshop on Information Hiding, pages 161–177. Springer, 2010.

[12] Yinlong Qian, Jing Dong, Wei Wang, and Tieniu Tan. Deep learning for steganalysis via convolutional neural networks. In SPIE/IS&T Electronic Imaging, pages 94090J–94090J. International Society for Optics and Photonics, 2015.

[13] Lionel Pibre, Jérôme Pasquet, Dino Ienco, and Marc Chaumont. Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover source mismatch. Electronic Imaging, 2016(8):1–11, 2016.

[14] Lionel Pibre, Pasquet Jérôme, Dino Ienco, and Marc Chaumont. Deep learning for steganalysis is better than a rich model with an ensemble classifier, and is natively robust to the cover source-mismatch. arXiv preprint arXiv:1511.04855, 2015.

[15] Sabah Husien and Haitham Badi. Artificial neural network for steganography. Neural Computing and Applications, 26(1):111–116, 2015.

[16] Imran Khan, Bhupendra Verma, Vijay K Chaudhari, and Ilyas Khan. Neural network based steganography algorithm for still images. In Emerging Trends in Robotics and Communication Technologies (INTERACT), 2010 International Conference on, pages 46–51. IEEE, 2010.

[17] V Kavitha and KS Easwarakumar. Neural based steganography. PRICAI 2004: Trends in Artificial Intelligence, pages 429–435, 2004.

[18] Alexandre Santos Brandao and David Calhau Jorge. Artificial neural networks applied to image steganography. IEEE Latin America Transactions, 14(3):1361–1366, 2016.

[19] Robert Jarušek, Eva Volna, and Martin Kotyrba. Neural network approach to image steganography techniques. In Mendel 2015, pages 317–327. Springer, 2015.

[20] Shumeet Baluja. *Hiding Images in Plain Sight: Deep Steganography*, (2019).

[21] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep

network with a local denoising criterion. Journal of Machine Learning Research, 11(Dec):3371–3408, 2010.

[22] Anthony J Bell and Terrence J Sejnowski. The “independent components” of natural scenes are edge filters. Vision research, 37(23):3327–3338, 1997.

[23] Wang Chung-Ming, Wu Nan-i, Tsai Chwei-shyong, et al. “A High Quality Steganographic Method with Pixel Value Differencing and Modulus Function,” Journal of Systems and Software, 2008, 81(1): 150-158.

[24] Wu Da-chun, Tsai Wen-Hsiang. “A Steganographic Method for Images by Pixel Value Differencing,” Pattern Recognition Letters, 2003, 24(9): 1613-1626.

[25] Zhang, Xinpeng and Shuozhong Wang. “Vulnerability of pixel-value differencing steganography to histogram analysis and modification for enhanced security.” Pattern Recognit. Lett. 25 (2004): 331-339.