SAPIENZA
UNIVERSITÀ DI ROMA

Neural Networks Final Project

Implementation of SteganoGAN: High Capacity Image Steganography for Image and Audio Input with GANs

Members:

Akash Garg
189271

Simone Rossetti
1900592

Supervisor:
Prof. Danilo Comminiello

Examiner:
Prof. Aurelio Uncini

Intelligent Signal Processing And Multimedia Lab
SAPIENZA UNIVERSITY OF ROME
Rome, Italy 2020

Neural Networks Final Project
Implementation of SteganoGAN: High Capacity ImageSteganography for Image and Audio Input with GANs
Akash Garg - Simone Rossetti
Intelligent Signal Processing And Multimedia Lab
SAPIENZA UNIVERSITY OF ROME
Rome, Italy 2020

Abstract

Steganography is a generic term that denotes all those techniques that somehow try to hide information within other forms of data. Differently from cryptography, which aims to hide messages by manipulation of the data, steganography aims to hide the existence of the information itself.
In this paper we are going to present steganography applied to images and audio, and in particular we will analyze the benefits that generative adversarial training produces in this context. The method used consists in three networks which works toghether: the first, which from now on we will refer to as *encoder*, responsible for hiding the information, the second, named *decoder*, responsible for recovering the secret message and the third, called the *critic* which detect the presence of the hided information. The real place in which the adversarial training takes place is between the encoder and the critic, this last one provides feedback on the performance of the second and ensure the encoder to produce realistic images as much as possible.
In this project we re-implemented and extended the work done by Kevin A. Zhang, Alfredo Cuesta-Infante, Lei Xu and Kalyan Veeramachaneni, in this paper: *SteganoGAN: High Capacity Image Steganography with GANs*, `https://arxiv.org/abs/1901.03892`, and by $DengpanYe$, $ShunzhiJiang$, and $JiaqinHuang$ in this paper *Heard More Than Heard: An Audio Steganography Method Based on GAN*, `https://arxiv.org/abs/1907.04986`

Keywords: Image Steganography, Audio Steganography, Information Hiding, Unsupervised Learning, Deep Learning, Generative Adversarial Network.

Acknowledgements

The authors would like to thank Prof. Aurelio Uncini whose lectures and seminars aroused our interest in the subject, and Prof. Danilo Comminiello, who supported, guided and directed us in this work.

Akash Garg, New Delhi, July 2020
Simone Rossetti, Roma, July 2020

Contents

List of Figures

List of Tables

1
Introduction

We can think about steganography as the science whose mechanisms allow two individuals, a sender and a receiver, to have a secret and private communication impossible to detect by human perception.
In this paper we are going to analyze a particular digital image steganography algorithm, which, in a kind of sense, implements an encryption algorithm too.
Infact, in this specific context we can identify two type of encryption: the first one, the more trivial, when the information hided is converted to bytes and then compressed. This data is not directly readable by anyone, apart from the two which hold the communication and the eventual interceptors which already know the exact decompression algorithm.
The second reason why this proposed algorithm is not just steganopgraphy but also encryption is that in this adversarial training process the encoder and decoder learn a huge amount of weights, and the networks themselves learn their own encryption and decryption methods, this means in practice that the networks are themselves the encryption and decryption keys and medium. This means in practice that the communication is completely private between the ones that hold the network models.
Massive surveillance operations have shown that the mere existence of meta-data communication could lead to privacy leakages even if the content is unknown. Therefore, steganography is necessary for private communication. Image steganography could be used for example in medicine to hide private patient information in images such as X-rays or MRIs as well as biometric data. Audio steganography could be used in the watermark, copyright protection and secret transmission and many other applications. Thus in general, in the media sphere, steganography can be used to embed copyright data and allow content access control systems to store and distribute digital works over the Internet.

1.1 Overview

The work is organized as follows: In Chapter 2 we are going to present the background of our work: the framework, the libraries and the datasets used. In Chapter 3 we will focus on the particular network architecture we adopted. Chapter 4 contains our experiments for several variants of the model. Chapter 5 is completely dedicated to the application of this architecture to audio data. In Chapter 6 we will discuss about some basic knowledge to evaluate the performances and the results obtained. Chapter 7 illustrates the results and Chapter 8 contains the conclusions, the possible future applications and summarizes our work.

(a) (b) (c) (d) (e)

Figure 1.1: (a) A randomly selected cover image; (b-c) steganographic image & difference image (between (a) & (b)) at data depth = 2; (d-e) steganographic image & difference image (between (a) & (d)) at data depth = 4

(a) (b) (c) (d) (e)

Figure 1.2: (a) A randomly selected cover image; (b-c) steganographic image & difference image (between (a) & (b)) at data depth = 4; (d-e) steganographic image & difference image (between (a) & (d)) at data depth = 8

2

Resources Used

2.1 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.

2.2 Datasets

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 total size of 3.5GB. On the other hand the validation set contains 200 images with total size of 450MB.

Instead for the audio experimets, we trained our models on the *FSDKaggle2018* dataset by Eduardo Fonseca, Manoj Plakal, Frederic Font, Daniel P. W. Ellis, Xavier Favory, Jordi Pons, Xavier Serra. This huge dataset contains 11,073 audio files, the majority are sounds of musical instruments. The train dataset contains about 9.5k with a total size of 5.69GB. Instead the validation dataset contains about 1.6k samples for a total size of 737.8MB. Obviously we decided to select randomly 800 files of train dataset and 200 of test dataset.

2.3 Framework and Libraries used

- **PyTorch:** We used PyTorch, the open source machine learning library, throughout the project. The library is primarily based and developed upon the Torch library. It provides easy to use functions and high level APIs much required for Machine learning and Deep Neural Network projects.

- **Librosa:** Librosa is a python package which is used for audio processing and analysis. We used this package primarily to obtain frequency spectrum of input audio files. The spectral images are used as cover to hide a secret message. The package also provides functionality to reconstruct audio from the steganographic spectrum images.

3

Architecture

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 message inside it; the decoder takes the image generate 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.

3.1 Encoder

The job of the encoder network is to take a cover image C and input a text which the user wants to hide. The text is converted into a binary data tensor $M \in \{0,1\}^{D \times W \times H}$. Here D is the number of bits that we wish to hide in each pixel of the cover image and $W\&H$ denote the width and the height of the cover image.

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:

$$a = Conv_{3 \to 32}(C) \tag{3.1}$$

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

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

3.1.1 Basic:

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

$$\mathcal{E}_b(C, M) = Conv_{32 \to 3}(Conv_{32 \to 32}(b)) \tag{3.3}$$

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:

$$\mathcal{E}_r(C, M) = C + \mathcal{E}_b(C, M) \tag{3.4}$$

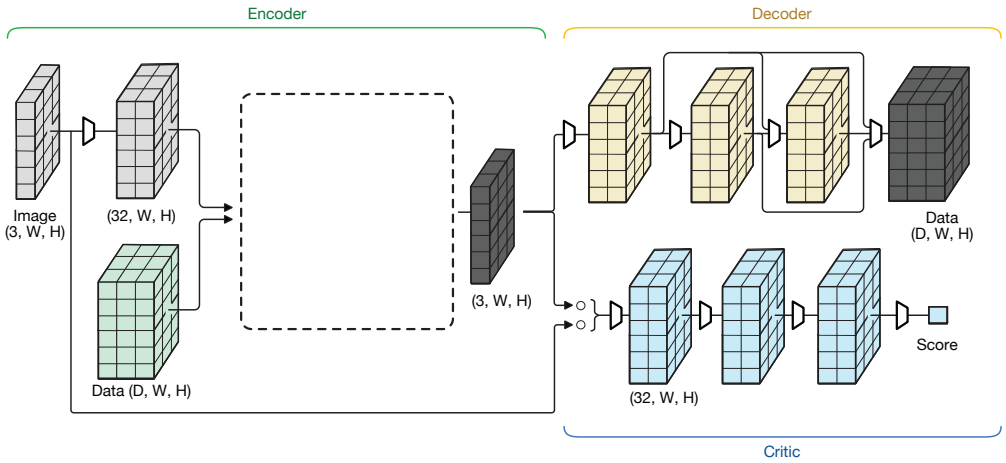


Figure 3.1: Overall architecture of the model including encoder, decoder and critic

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:

$$\begin{cases} c = Conv_{64+D\to 32}(Cat(a, b, M)) \\ d = Conv_{96+D\to 3}(Cat(a, b, c, M)) \\ \mathcal{E}_d(C, M) = C + d \end{cases} \tag{3.5}$$

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

3.2 Decoder

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

$$\begin{cases} a = Conv_{3\to32}(S) \\ b = Conv_{32\to32}(a) \\ c = Conv_{64\to32}(Cat(a,b)) \\ \mathcal{D}(S) = Conv_{96\to D}(Cat(a,b,c)) \end{cases} \tag{3.6}$$

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:

$$\begin{cases} a = Conv_{32\to32}(Conv_{32\to32}(Conv_{3\to32}(S))) \\ \mathcal{C}(S) = Mean(Conv_{32\to1}(a)) \end{cases} \tag{3.7}$$

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

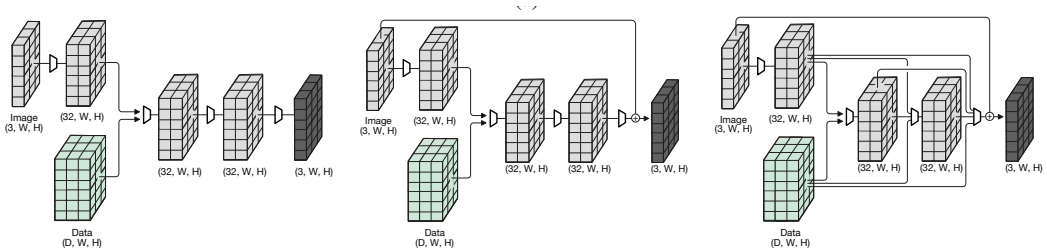


Figure 3.2: The three variants of Encoder, Basic (left); Residual (middle); Dense (right)

4

Training

Generative Adversarial Networks (GANs) 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.

4.1 Batch Size

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. Considering the size of the dataset we chose to train our model on we took a small batch size of 4. The batch size was quite suitable for our case and we observed that both the decoder as well as the encoder could train with stability.

4.2 Transforms of Training Samples

Thanks to *torchvision.transforms* we could chain together a number of image transforms before feeding the training samples to the network for training. Below we list the transforms used:

- **Random crop:** This helps to crop the image at a random location.
- **Random horizontal flip:** This flips the given image randomly.
- **Normalize:** Using this function we normalize the given image to have unit mean as well as unit standard deviation.

4.3 Loss Functions

In order to generate good results we need to optimize the three networks involved namely, encoder, decoder and critic. We perform the optimization iteratively. The optimization of the encoder-decoder network is done by optimizing three losses simultaneously. These are:

- decoder loss:

$$\mathcal{L}_d = \mathbb{E}_{X \sim P_c} CrossEntropy(\mathcal{D}(\mathcal{E}(X, M)), M) \tag{4.1}$$

- mean square error:

$$\mathcal{L}_s = \mathbb{E}_{X \sim P_c} \frac{1}{3 \times W \times H} \|X - \mathcal{E}(X, M)\|_2^2 \tag{4.2}$$

- realness score of steganographic image:

$$\mathcal{L}_r = \mathbb{E}_{X \sim P_c} \mathcal{C}(\mathcal{E}(X, M)) \tag{4.3}$$

Now, while training the objective function is:

$$minimize \quad \mathcal{L}_d + \mathcal{L}_s + \mathcal{L}_r \tag{4.4}$$

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

$$\mathcal{L}_\mathcal{C} = \mathbb{E}_{X \sim P_c} \mathcal{C}(X) - \mathbb{E}_{X \sim P_c} \mathcal{C}(\mathcal{E}(X, M)) \tag{4.5}$$

5

Audio Spectrum Handling

To deal with audio steganography we decided to pass through spectrogram representation of the audio to exploit the already present architecture used in image steganography.

5.1 Direct and inverse problem

Similarly to image steganogrpahy, the problem is formulated as follows:

- *Direct problem:* an audio, transformed into spectrogram, and a secret message (payload) are given as input to an *encoder* which hide the message into the frequencies of the audio. Here the spectrogram with containing the hided message is converted back into audio. Steganographic audio is made.
- *Inverse problem:* to get back and decode the secret message, the audio needs to be transformed again into spectrogram and the *decoder* will extract the secret message.

The *critic* still plays the same role: it judges the quality of the encoder to improve its results. Obviously both problems are not so trivial as one would expect.

5.2 Experiments

We encountered a lot of limitations dealing with frequencies, given by mathematical approximations which we wanted to avoid, since approximate the result of the conversion audio-spectrogram deteriorate too much the original audio and destroy the hidden message.

In the following experiments our purpose has been to not alter excessively the network architecture. The audio files used were not subject to any type of transformation, except for the Fourier transformations. Thus spectrograms have not been altered by rotations or translations.

5.2.1 Spectrogram and Griffin-Lim algorithm

This strategy was inefficient, and we will try to explain the reasons.
Spectrogram is a visual representation of the spectrum (power) of frequencies of a signal as it varies with time. In practice the power spectral density of a signal is computed as the square of the absolute value of the $STFT$, which is a non linear operation, infact to deal with this, it's inverse can be computed thanks to an iterative

algorithm which tries to guess and recover the phase of the original audio, this is called Griffin-Lim algorithm.
Choosing this strategy we could preserve the already existent networks structures. We can assert this method downgraded the quality of the sounds although they were still audible, but was unmanageable from the steganography point of view, infact Griffin-Lim approximation destroyed the hidden information.

5.2.2 STFT and ISTFT

This strategy instead gave us very good results. We used as guideline the work done by *DengpanYe*, *ShunzhiJiang*, and *JiaqinHuang* in this paper *Heard More Than Heard: An Audio Steganography Method Based on GAN*, `https://arxiv.org/abs/1907.04986`. Which work differs from our in the fact that they hide a secret audio inside a carrier one and the decoder learns to recover the hidden audio. Instead, in our work, we are trying to handle two different type of information, audio and text, that's why we encountered a lot of decoding problems.

Each audio was re-sampled at $5kHz$ or $22kHz$ (depending on the dataset) and, this time, we stopped one step before and converted the signal by short-time fourier transform $STFT$ with 719 FFT window size to ensure 360 time frames, and variable hop window to reduce frequency frames to 360 samples. This, in order to have $STFT$ of dimensions $(360, 360)$. The audio after $STFT$ was complex and in the form of $a + ib$. In order to train the networks, it needs to be expressed as real numbers. There were two strategies to spilt $a + ib$ in two matrices: divide it in real part and imaginary part, or amplitude and phase part. Considering that the magnitude and phase have different dimensions which can not be directly overlapped and processed at the same time, the first strategy was chosen. The complex number was divided into a real part and imaginary part: extract the real coefficients a and b, then overlapped them to produced a two-channel matrix. In the inverse transformation, the two matrices are merged into a complex number matrix $a+ib$ and decoded.

In this way we modified the precedent architectures, the encoder now takes as input 2 channels $STFT$ matrix and a payload and produces a 2 channels $STFT$ matrix. The decoder and the critic have as input a 2 channels $STFT$ matrix too, and the output is unchanged.

6

Evaluation Metrics

Steganography algorithms are evaluated along three aspects:

- *capacity*: the amount of data that can be hidden in an image/audio,
- *distortion*: the similarity between the cover and steganography image/audio,
- *secrecy*: and 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.

6.1 Reed Solomon bits-per-pixel (RSBPP)

First of all to guarantee the effectiveness of the algorithm with respect to the *capacity* we want to measure the effective number of bits that can be conveyed per pixel. Someone could simply think that this number is proportional to the amount of pixels, but the problem is not as trivial it may appear, since the ability to recover a hidden bit is heavily dependent on the model and the cover image, as well as the message itself.
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 such trivial, thus, we have to introduce a mechanism to recover the information hided 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 and 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 $\frac{n-k}{2} = \frac{m}{2}$ 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:

$$p \cdot n \leq \frac{n-k}{2} \Longrightarrow \frac{k}{n} \leq 1 - 2p \tag{6.1}$$

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.

6.2 Peak Signal to 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):

$$MSE = \frac{1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} (X_{i,j} - Y_{i,j})^2 \tag{6.2}$$

$$PSNR = 20 \cdot \log_{10}(\gamma) - 10 \cdot \log_{10}(MSE) \tag{6.3}$$

6.3 Structural Similarity Index (SSIM)

Structural similarity index is a second *similarity* metrics 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 covariance σ_{XY}^2

$$SSIM = \frac{(2\mu_X\mu_Y + k_1R)(2\sigma_{XY}^2 + k_2R)}{(\mu_X^2 + \mu_Y^2 + k_1R)(\sigma_X^2 + \sigma_Y^2 + k_2R)} \tag{6.4}$$

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.

7
Results

In this section we discuss about the various experiments conducted with different models that we trained for the completion of the task. Below in section:7.1 we analyse the results obtained when images were used as the direct input to various models trained, whereas in section:7.2 results are presented for the situation where audios were the direct input to the algorithm.

7.1 Image

As discussed in section:2.2 we used Div2k dataset for the training and validation of the networks developed for the fulfillment of the project. Every trained model is validated and the relevant parameters namely, bits per pixel, peak signal to noise ratio, structural similarity index, stability of encoder and decoder networks were recorder for analysis.

7.1.1 Comparison at different data depths

For a simple comparison we select four models out of all the trained set 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 model is provided with a sequence of image 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 : 7.1. The table lists the values of the parameters discussed in Chapter:6.

Performance on Dense Encoder and Dense Decoder				
D	RS-BPP	PSNR	SSIM	Decoder Accuracy
2	1.90	38.01	0.89	0.98
4	2.39	37.34	0.87	0.80
6	2.04	38.71	0.93	0.67
8	2.19	39.69	0.93	0.63

Table 7.1: Performance of various SteganoGAN models on validation dataset each trained for 32 epochs at different data depths.

From the table 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. Moreover, we try to provide an intuition of visual difference between the models trained at various data depths in Figure:7.1.

Figure 7.1: A pair of cover (left) and steganographic images (right) and the difference between them. (a-c) for D=2; (d-f) for D=4; (g-i) for D=6; (j-l) for D=8

Figure:7.1 shows that the similarity between the cover image and the steganograohic image is highest for model trained for $d = 2$. However an interesting fact here is that the similarity is more for model trained at data depths 6 & 8 than that trained at 4. This is because effectively model at data depth = 4 hides more bits per pixel compared to the other three models. Therefore, in view of the above discussion we chose to keep data depth = 4 for further development and analysis.

7.1.2 Comparison at different epochs

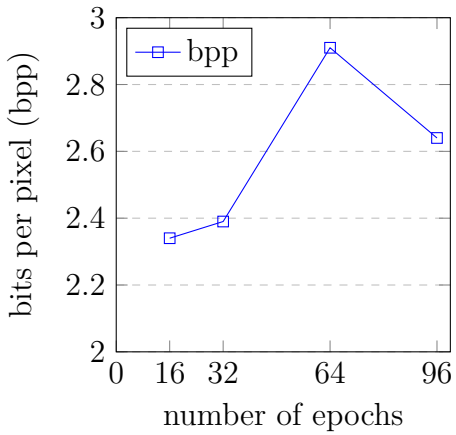
After the analysis that a model trained for data depth value equal to 4 outperforms other models we did an evaluation of training models at different epochs for $d = 4$. We selected four different values of epochs, $16, 32, 64 \,\&\, 96$, for comparison. Figure:7.2 displays the values of various evaluation metrics at the last epoch on validation set for each of the four models.
As evident from the figure:7.2 *bpp* is highest with the value of 2.91 for a model trained for 64 epochs, however the value decreases if the epochs are increased to 96. Similarly the *encoder and decoder losses* are the least with the values 0.00061 and 0.27, respectively, for the same model.
It is quite tempting to train the GAN for as long as possible within the bounds of available resources, but we observed that over-training actually degraded the quality of the model as well as the generated steganographic images.

7.2 Audio

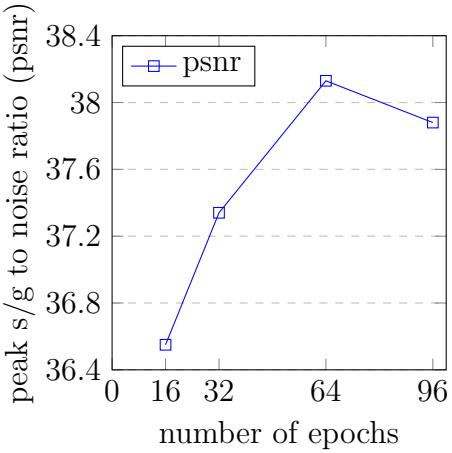
We decided to train the network on *FSDKaggle2018* which is made by very short audio segments, sampled at 22kHz.
In order to generate 360×360 spectrograms, audio files hes been sampled with 719 FFT window size and variable hop length.
We trained new models for dealing with 2 channels input size as explained in Subsection 5.2.2.

7.2.1 Comparison at different data depths

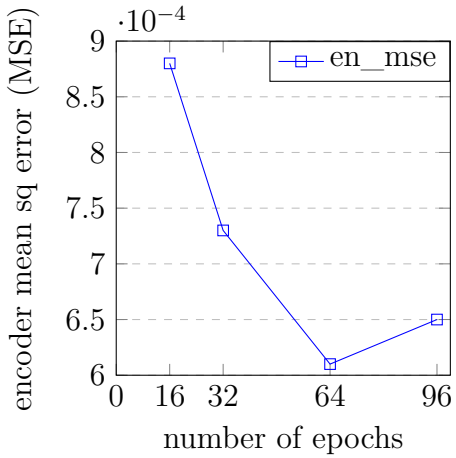
We repeated the tests made for image steganography using second strategy discussed in Subsection 5.2.2. This strategy permitted us to perfectly detect the message hided into STFT and perfectly reconstruct the original audio. We selected three models out of all the trained set of models. Each of the four selected models were trained for 32 epochs, this time at data depths 2, 4 and 6. The results are shown in Table: 7.2.
From the table it can be seen that again the model trained for data depth = 4 achieves a greatest *RS-BPP*. We can see the characteristics of the network are quite preserved with respect to the Table: 7.1. We can visualize again the difference between the models trained at various data depths in Figure:7.3. We can visualize how these three models in Figure:7.3, each trained respectively with 2,4 and 6 depth, learn to hide data in a very different strategies. We can clearly see how model trained with depth = 4 hides the secret message where pixels have lower values (in the black



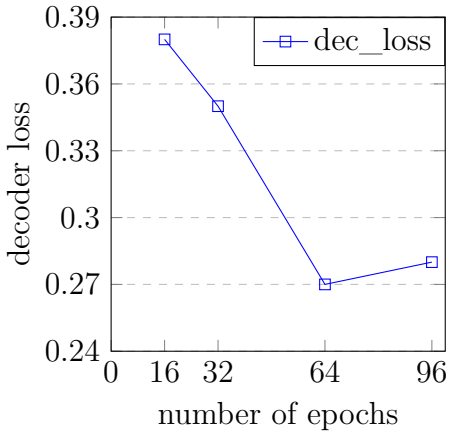
(a) bpp vs number of epochs representation



(b) psnr vs number of epochs for data representation



(c) encoder mse vs number of epochs representation



(d) decoder loss vs number of epochs for data representation

Figure 7.2: Performance of SteganoGAN model on validation dataset trained for data depth=4 at different epoch numbers

part). Instead the model with depth = 6 learned to hide the data where pixels have higher values.

We chose again to keep data depth = 4 for further development and analysis.

7.2.2 Comparison at different epochs

Again we performed the same analysis for the model with depth value equal to 4 at different epochs. We selected the same four different values of epochs. Figure:7.4 displays the values of various evaluation metrics at the last epoch on validation set. We observe again from Figure:7.4 that over-training actually degraded the quality of the model as well as the generated steganographic images.

Performance on Dense Encoder and Dense Decoder				
D	RS-BPP	PSNR	SSIM	Decoder Accuracy
2	1.82	39.47	0.82	0.96
4	2.26	42.55	0.84	0.78
6	2.23	42.32	0.81	0.68

Table 7.2: Performance of various audio adapted SteganoGAN models on validation dataset each trained for 32 epochs at different data depths.

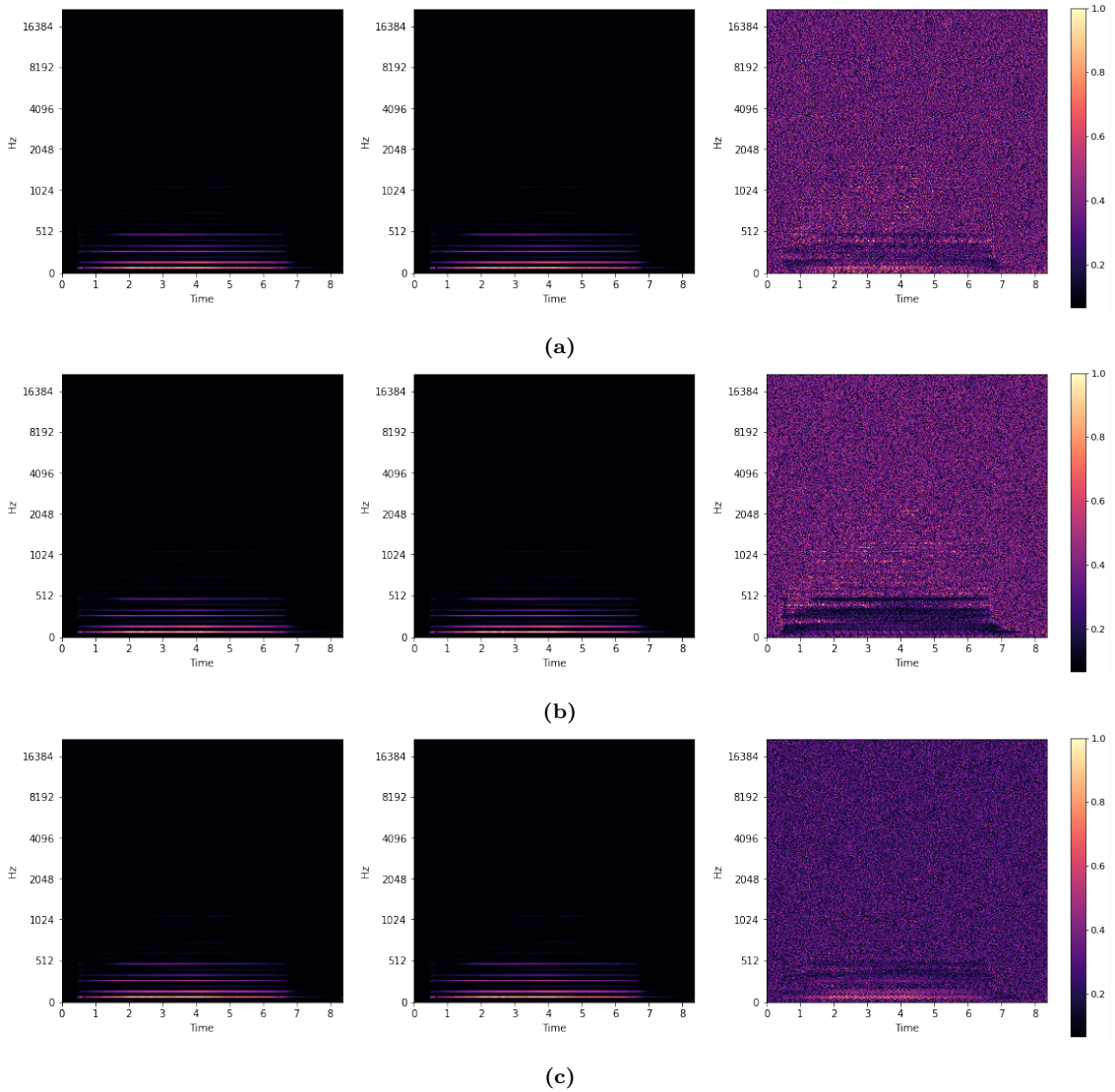
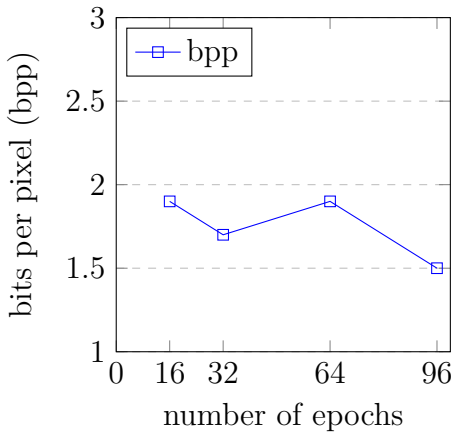
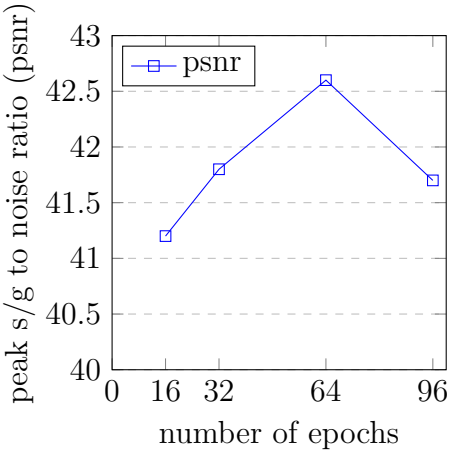


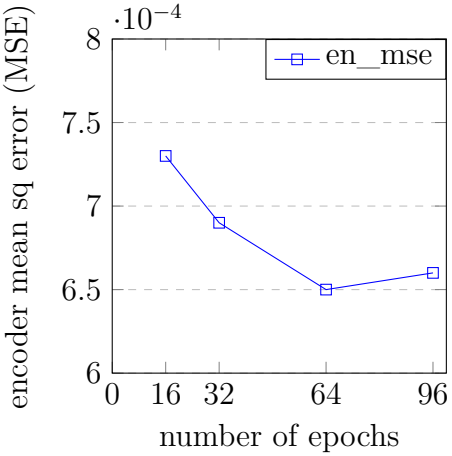
Figure 7.3: STFT of a 'LA' (A) sound of a violin: (left) cover, (center) steganographic and their difference, the generated payload (right). Row (a) is for D=2; (b) for D=4; (c) for D=6.



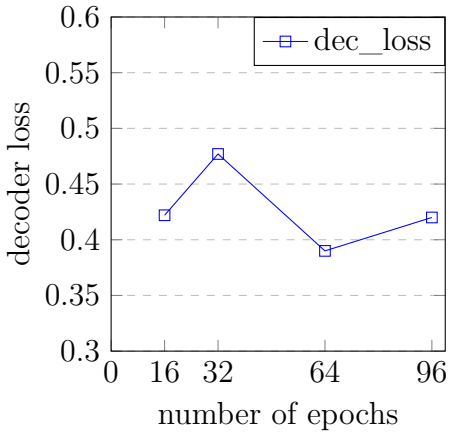
(a) bpp vs number of epochs representation



(b) psnr vs number of epochs for data representation



(c) encoder mse vs number of epochs representation



(d) decoder loss vs number of epochs for data representation

Figure 7.4: Performance of audio adapted SteganoGAN model on validation dataset trained for data depth=4 at different epoch numbers

8

Conclusion

In this paper, we demonstrated the flexibility of this new approach to steganography which can be applicable to any kind of data and that can hide any kind of information. Furthermore, we analyzed in detail new different metrics for evaluating the performance of deep-learning based steganographic systems so that they can be directly compared against traditional steganography algorithms.
We experiment with the three variants of the *STEGANOGAN* architecture and evaluate the high capacity of the model to achieve higher relative payloads than existing approaches while still evading detection. We would like to further investigate generative adversarial approaches to steganography exploring new possible architectures and consolidate a new approach to audio encoding and decoding.