International Journal For Technological Research In Engineering
Volume 7, Issue 12, August-2020 *ISSN (Online): 2347 - 4718*

COVER IMAGE MODIFICATION IN IMAGE STEGANOGRAPHY USING GENERATIVE ADVERSARIAL NETWORKS

Abhijay Kumar Singh[1], Prof. Swetha P. M[2], Manish R. Jain[3], B. Akash[4]
[1, 3, 4]BE students, [2]Assitant Professor,
Department of Computer Science and Engineering, JSS Science and Technology University, Mysore-570006

Abstract:* Steganography is the practice of concealing a secret message within another, ordinary message. Commonly, steganography is used to unobtrusively hide a small message within the noisy regions of a larger image. But with the recent advances in steganalysis techniques using deep neural networks, attackers are able to breach covert communication by identifying regions with distortions in the stego-media. Generative Adversarial Networks (GANs) have achieved many research results in the field of computer vision and have also demonstrated great potential in image steganography. In this paper, we attempt to modify cover images used in steganography by generating more suitable and secure cover images. The proposed model employs a generative network and two discriminative networks. The generative network is utilized to generate the original carrier information with noise as the driver and to evaluate the visual quality of the generated cover images. The discriminative networks are utilized to assess the suitableness of the generated images for information hiding. A sophisticated steganalysis network is used as an extra discriminative network to ensure that the generated carriers are immune to steganalysis techniques.

***Keywords:* Steganography; Generative Adversarial Networks; Steganalysis**

I. INTRODUCTION

With the rise of cyber-crime and data breaches, the need to hide confidential information and achieve secure communication is a challenge. Image steganography is dedicated to hiding secret messages in digital images and has achieved the purpose of covert communication and secret data storage. In modern steganography, the most common steganography technique based on different construction strategies of stego-images is cover image modification. Cover image modification steganography achieves the purpose of hiding messages by making use of redundant information in the digital carrier and modifying this redundant information. There exist two mainstream approaches for cover modification steganography: One approach is to maintain the invariance of a statistical model and the other type of methods implement embedding by minimizing a certain distortion function. However, the modification will inevitably lead to the difference in the distribution between the original cover and the stego carrier. Steganalysis has been extensively studied in the last decade. Its main purpose is to detect the presence of secret messages in digital covers such as digital images coming from a known source. Usually, this task is formulated as a binary classification problem to distinguish between cover and stego objects. Conventionally, the steganography methods heuristically consider the steganalysis side, e.g. the message should embed into the noise and texture region of image which is more secure. These loosely defined rules tend to make generated stego-images to be susceptible to steganalysis.

The promise of deep learning is to discover rich, hierarchical models that represent probability distributions over the kinds of data encountered in artificial intelligence applications, such as natural images, audio waveforms containing speech and symbols in natural language corpora. So far, the most striking success has involved discriminative models, usually that map a high-dimensional, rich sensory input to a class label. As a framework of generative model, Generative Adversarial Network (GAN), proposed in 2014, is able to generate better synthetic images than previous generative models, and since then it has become one of the most popular research areas. Recently, several articles using GAN to design steganographic methods have enriched the technical means of steganography.

In this paper, we implement a model that employs a Generative Adversarial Network to generate more suitable cover images that improve the security of steganography. Using a WGAN, we are able to achieve high visual quality in the generative images and also a faster training process. With the use of a steganalysis network in the adversarial learning between the generator and discriminators, the generated images are more suitable for embedding in the steganography process.

II. LITERATURE SURVEY

The name “GAN” was introduced by Ian Goodfellow et al [1] in 2014. Their paper popularized the concept and influenced subsequent work. It introduced a new framework for estimating generative models via an adversarial process, in which they simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. There was no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrated the potential of the framework through qualitative and quantitative evaluation of the generated samples.

Radford et al [4] proposed the most famous GAN, the

DCGAN, which adopted the use of a Convolutional Neural Network (CNN), a model predominantly used in supervised learning, in an unsupervised learning scenario. This model performs well in image synthesis in the early work of the research. In order to control generated result, different GAN such as CGAN, InfoGAN, and ACGAN are proposed. Some methods have been proposed for solving the model collapse problem by designing a new loss function such as mini-patch feature and WGAN.

Training GANs requires finding a Nash equilibrium of a non-convex game with continuous, high-dimensional parameters. GANs are typically trained using gradient descent techniques that are designed to find a low value of a cost function, rather than to find the Nash equilibrium of a game. Salimans et al [3] proposed a variety of architectural features and training procedures to the GAN framework. A few findings by the authors for convergence in training a GAN include adopting a robust feature matching approach that can address the instability of the GANs by specifying a new objective for the generator that prevents it from overtraining on the current discriminator. Instead of directly maximizing the output of the discriminator, the new objective requires the generator to generate the data that matches the statistics of the real data, where we use the discriminator only to specify the statistics that we think are worth matching. One of the main failure modes for GAN is for the generator to collapse to a parameter setting where it always emits the same point. When collapse to a single mode is imminent, the gradient of the discriminator may point in similar directions for many similar points. Because the discriminator processes each example independently, there is no coordination between its gradients, and thus no mechanism to tell the outputs of the generator to become more dissimilar to each other. To overcome this failure mode during the training process, the authors suggest adopting mini-batch discrimination. They also proposed adopting batch normalization and strided convolutions to achieve greater accuracy while training GANs.

The use of GANs in image steganography is a relatively new research area. Cover modification by GAN focuses on the adversarial game between steganography and steganalysis. These methods use generator trained by GAN to construct various core elements in the cover modification scheme. One strategy is to generate the original cover image, the second is to generate the modification probability matrix in the framework of minimizing distortion, and the third is to directly use the adversarial game among tripartite to learn a modification algorithm for generating stego images.

Volkhonskiy et al [2] proposed the application of GAN to steganography. They constructed a special generator for cover-image creation. Synthetic images generated by this generator were less susceptible to steganalysis compared to covers. This approach allowed generating more steganalysis-secure cover that could carry message using standard steganography algorithms such as LSB or STC. They introduced the Steganographic Generative Adversarial Networks called SGAN, which consisted of three networks. A generator G, a discriminator D and a steganalysis classifier S which determined if a realistic image was hiding secret information. SGAN trains G with D and S simultaneously.

Similar to SGAN proposed by [2], Shi et al [7] use the same strategy that generates cover images for steganography with adversarial learning scheme, named SSGAN. The SSGAN architecture also has one generative network called G, and two discriminative networks called D and S. A more complex network called GNCNN [6] is chosen as the discriminator D and the steganalyser S.

Another interesting cover image generation method is proposed by Wang et al [9]. Unlike SGAN and SSGAN, a discriminator D determines whether an image is stego or real image, stego(G(z)) and real image sample x are used as the input of discriminator model D. Such improvement makes the distribution of the stego image closer to the real data distribution. An interesting result of this scheme is that the images generated directly by the generator may not be realistic, and the fidelity of the stego image is achieved after the modification operation.

Ni et al [5] introduce hierarchical representations for deep learning in image steganalysis. Preceding this paper, the prevailing detectors of steganographic communication in digital images mainly consisted of three steps, i.e., residual computation, feature extraction, and binary classification. They introduced an alternative approach to steganalysis of digital images based on convolutional neural network (CNN), which is shown to be able to well replicate and optimize these key steps in a unified framework and learn hierarchical representations directly from raw images. The proposed CNN has a quite different structure from the ones used in conventional computer vision tasks. Rather than a random strategy, the weights in the first layer of the proposed CNN were initialized with the basic high-pass filter set used in the calculation of residual maps in a spatial rich model (SRM), which acted as a regularizer to suppress the image content effectively. To better capture the structure of embedding signals, which usually have extremely low SNR (stego signal to image content), a new activation function called a truncated linear unit was adopted in their CNN model. Finally, they further boosted the performance of the proposed CNN-based steganalyzer by incorporating the knowledge of selection channel.

III. PROPOSED METHOD

This section describes the framework and the details of our proposed method. Firstly, we introduce the adversarial learning procedure. Then we elaborate the architecture and the loss function designed specifically for the three networks. The implementation details are given in the last subsection.

A. Adversarial Learning

Adversarial learning of the model employs an unsupervised approach coupled with the Zero-Sum concept from game theory to reach a Nash Equilibrium. The independent generator and discriminator models are trained to compete with each other; iteratively, improving the output of each model. In Generative Adversarial Networks, the generative model G tries to learn to map the noise to samples, while the discriminative model D tries to distinguish between the sample outputs by G and the real samples. Based on fooling the discriminator D, the weight of the generator G is updated, and at the same time, the discriminator D's weight is updated by distinguishing between the fake and real samples. In recent years, GANs have been successfully applied to image generation tasks using convolutional neural networks (CNNs). But traditionally, GANs are considered difficult to train due to an unclear definition of the convergence between the loss function and the sample outputs quality. Typically, people choose to stop the training by visually checking the generated samples. To overcome this shortcoming, WGAN [10], using the Wasserstein distance instead of the Jensen-Shannon divergence was proposed to define the convergence criteria between the data set distribution and the learning distribution of the generator G. Evidently, the sample quality was closely related with the network's convergence. The rate of training of the network also improved drastically.

The adversarial training process can be described as a minimax game:

$$\min_G \max_D J(D,G) = E_{x\sim p_{data}(x)} \log\big(D(x)\big) + E_{z\sim p_{noise}(z)} \log(1 - D(G(z)$$

, where *D(x)* represents the probability that *x* is a real image rather than synthetic, and *G(z)* is a synthetic image for input noise *z*. In this process, the two networks, the generator G and the discriminator D are trained simultaneously:

- The generative network takes in as input a vector ***z*** from the prior distribution $p_{noise}(z)$ and transforms it into a meaningful output from the data distribution $p_{data}(x)$ trying to make the generated image similar to $p_{data}(x)$.
- Mathematically, a discriminative model takes an input x that can be a real one (x_t, whose density is denoted p_t) or a generated one (x_g, whose density p_g is the density induced by the density p_g going through G) and that returns the probability *D(x)* of *x* to be a real data. Discriminator D is run twice, once with the real sample and once with the generated sample. Losses incurred during both these runs are used to calculate the total loss associated.
- To solve the minimax problem, in each iteration of the mini-batch stochastic gradient optimization, we first perform the gradient ascent step on D and then perform the gradient descent step on G. If we assume ω_N to represent the neural network N, then we can see that the optimization steps are as follow:
  i. We let D fixed to update the model G by $\omega_G \leftarrow \omega_G - \gamma_G \nabla_G J$, where:

$$\nabla_G J = \frac{\partial}{\partial \omega_G} E_{z\sim p_{noise}(z)} \log(1 - D(G(z,\omega_G),\omega_D))$$

  ii. We let G fixed to update the model D by $\omega_G \leftarrow \omega_G - \gamma_G \nabla_G J$.

$$\nabla_D J = \frac{\partial}{\partial \omega_D}\{E_{x\sim p_{data}(x)} \log\big(D(x,\omega_D)\big) + E_{z\sim p_{noise}(z)} \log\big(1 - D(G(z,\omega_G),\omega_D$$

B. *System Architecture Overview*

The generative network G is designed with one fully connected layer, four fractionally strided layers and one hyperbolic tangent function layer. The discriminative network D (Figure 1) comprises four layers, one fully connected layer. As for the steganalyser network S (Figure 2), we use a predefined high-pass filter to perform a filtering operation, which is kept fixed while training, and four convolutional layers corresponding to three kinds of operations, i.e. convolution, non-linearity, and pooling, followed by a fully connected layer and a softmax layer for classification.

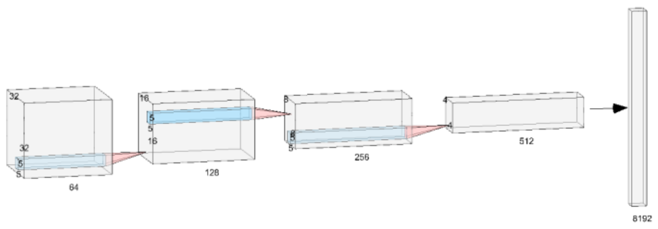

Figure 1: Discriminator D

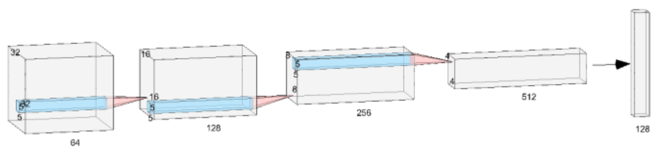

Figure 2: Discriminator S

As the generator improves with training, the discriminator performance gets worse because the discriminator can't easily tell the difference between real and fake. If the generator succeeds perfectly, then the discriminator has a 50% accuracy. In effect, the discriminator flips a coin to make its prediction.
Core to this design is adopting and modifying three recently demonstrated changes to CNN architectures. They are discussed in the following subsections.

Strided Convolutions

In all the convolutional networks, the deterministic spatial pooling functions (such as maxpooling) are replaced with strided convolutions. This allows the network to learn from its own downsampled space. This approach is used in the generator, allowing it to learn from its own upsampled space.

Eliminate Fully Connected Layers

Elimination of fully connected layers on top of convolution layers has been showcased as a strong example in global average pooling. This can be clearly observed in state of the art image classification models. It was found that global average pooling increased model stability but hurt convergence speed. A middle ground of directly connecting the highest convolutional features to the input and output respectively of the generator and discriminator worked well. The first layer of the GAN, which takes a uniform noise distribution Z as input, could be called fully connected as it is just a matrix multiplication, but the result is reshaped into a 4-dimensional tensor and used as the start of the convolution stack. For the discriminator, the last convolution layer is flattened and then fed into a single sigmoid output.

Batch Normalization

Batch Normalization stabilizes learning by normalizing the input of each unit to have zero mean and unit variance. This helps deal with training problems that arise due to poor initialization and helps gradient flow in deeper models. Directly applying batch-norm to all layers however, resulted in sample oscillation and model instability. This was avoided by not applying batch-norm to the generator output layer and the discriminator input layer. The Leaky Rectified Linear Unit (LeakyReLU) activation is used in the generator with the exception of the output layer which uses the tanh (hyperbolic tangent) function. It was observed that using a bounded activation allowed the model to learn more quickly to saturate and cover the color space of the training distribution. Within the discriminator it was found that the leaky rectified activation to work well, especially for higher resolution modeling.

C. System Implementation Overview

Dataset Preparation

"The Simpsons Character dataset" from Kaggle is used in our experiments. The dataset of approximately 9000 images was used for training this model. All the images are cropped to 64x64 images. For the purpose of steganalysis, 90% of the data is used to construct a training set, and the rest is regarded as testing set. The training set is denoted by ***train***, and the testing set is denoted by *test*. We use Stego(x) to represent the steganographic algorithm used to hide information. Using the aforementioned information, two datasets are created that are utilized in the experiments. One is *train* + Stego(*train*), where Stego(*train*) is the training set embedded in some secret information, and the other is *test* + Stego(*test*).
Finally, we got 16,200 images for steganography training, 1,800 for testing. For embedding images with a secret message, we use the LSB Matching algorithm, a 1-embedding algorithm, with a payload size of 0.4 bits per pixel to embed information. The text for this is from a sample text document of 100 lines.

Loss Function

The model is trained with mini-batch stochastic gradient descent with a mini-batch size of 64. The learning rates are as follows:

a. for the generator G is set to 4 * 10 $^{-4}$
b. for the discriminator D is set to 4 * 10^{-5}
c. and for the discriminator S is 5 * 10^{-6}

The network utilizes the binary cross entropy loss function during the training of the steganalyzer.

Optimization

Due to a computational constraint and having a comparatively small dataset we have had to use many tricks to train a stable model with given conditions. While previous GAN work have used momentum to accelerate training, this model has incorporated the Adam optimizer. The update parameters between the generator and the two discriminators are as follows.
a. The update parameters for the Adam Optimizer β_1 and β_2 for optimizing the error between the generator G and discriminator D are initialized to 0.5 and 0.99.
b. Similarly, the optimizing parameters, for the Adam Optimizer for reducing the error between generator G and discriminator S, β_1 and β_2 is 0.9 0.99 respectively.

Model training

Since the dataset size is quite large, feeding the network with individual image inputs would not be the best approach as this would drastically increase the training period. To resolve this issue, batches of images are created. Batch size associated with each input is 64 images. No pre-processing was applied to training images besides cropping and scaling to the range of the tanh activation function [-1, 1]. All weights used in the network are initialized from a zero-centered Normal Distribution with standard deviation of 0.02. LeakyReLU function is used as activation between each convolutional layer. The LeakyReLU has slope of the leak set to 0.2 in all models. We update the weights of D and S once, while update weights of G twice in each mini-batch. We want the Generator G to generate realistic images that could be used as secure covers for steganography, we force G to compete against the discriminators D and S at the same time. We use S(x) to represent the output of the steganalysis network, then the game can be shown as follows:

$$\min_{G}\max_{D}\max_{S} J = \alpha\left(E_{x\sim p_{data}(x)}\log\left(D(x)\right) + E_{z\sim p_{noise}(z)}\log\left(1 - D\left(G(z)\right)\right)\right)$$
$$+(1-\alpha)E_{z\sim p_{noise}(z)}\left[\log S\left(Stego\left(G(z)\right)\right) + \log\left(1 - S\left(G(z)\right)\right)\right]$$

IV. TESTING AND ANALYSIS

The GAN after training is able to capture the features of the face such as the eyes, nose and hair of the characters as observed in most of the images generated. A few samples of the images generated by our model can be seen in the following Figure (3). The generator and discriminator losses are also captured in the Figure (4).

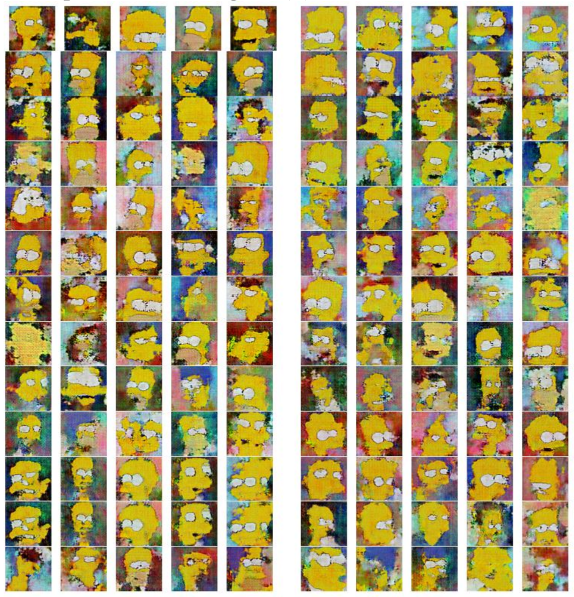
Figure 3: Examples of images generated by our GAN after training for 100 epochs.

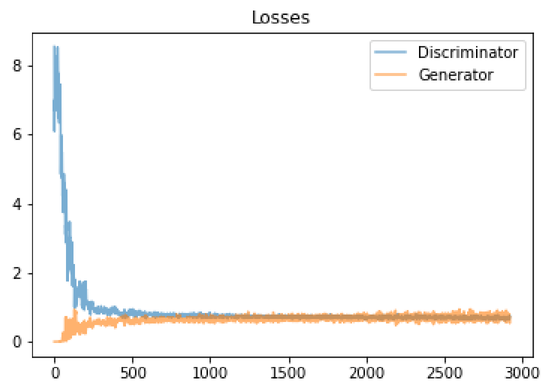

Figure 4: Losses observed in the Discriminator (*D*) and Generator (*G*).

V. CONCLUSION AND FUTURE WORK

In this work, we have implemented a Generative Adversarial Network for steganography to generate more suitable and steganalysis-secure covers for image steganography. We have successfully constructed a special cover-image generator that can use standard steganography algorithms such as LSB for image hiding. The approach focused on the adversarial game between steganography and steganalysis. A steganalyzer had been introduced against the steganography explicitly. This allowed for the generator to be more robust and generate more suitable and secure images that are able to fool the steganalyser. The implemented architecture is able to generate images of medium visual quality.

In our future work, we look to test the images generated by our network by embedding them with messages using the same algorithm as the one used to train the steganalyzer and then feeding the, into the steganalyser to test the robustness of the model. This can be used to estimate the classification error rates of the steganalyzer as well as the security of the images generated by the generator of our model. Another experiment that we can use to investigate the security of the images generated by our model can be to use different seed values. For this, we can use a pre-trained steganalyzer network.

We conclude by noting the robustness of the implemented model and the high quality of the images generated. This model has potential to be employed in cover image modification for applications in a field like military covert communication. The model has a long way to go before they can be trained on huge datasets that have much more apparent multimodality and features. But the field of machine learning is doing very well with the ever decreasing cost of computational power and ever increasing brain power invested in its research.

ACKNOWLEDGMENT

Firstly, we would like to express our sincere gratitude to Dr. M. P. Pushpalatha, HOD of Dept. of Computer Science and Engineering, JSS STU for providing an excellent environment for our education and encouraging us throughout our stay in college. We extend our heartfelt gratitude to our guide Prof. Swetha P. M, Assistant Professor of Dept. of Computer Science and Engineering, JSS STU who has supported us throughout our project with her patience and knowledge whilst allowing us the room to work in our own way.

REFERENCES

[1] Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Ward Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In NIPS, 2014.
[2] Denis Volkhonskiy, Boris Borisenko and Burnaev, Evgeny. Generative adversarial networks for image steganography. ICLR 2016 Open Review, 2016.
[3] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen. Improved Techniques for Training GANs. Computer Science, Mathematics NIPS 2016.
[4] Radford, Alec, Metz, Luke, and Chintala, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. CoRR, abs/1511.06434, 2015.
[5] Ni, J., Y. Jian, and Y.I. Yang, Deep Learning Hierarchical Representations for Image Steganalysis. IEEE Transactions on Information Forensics Security, 2017. 12(11): p. 1-1.
[6] Yinlong Qian, Jing Dong, Wei Wang, and Tieniu Tan. Deep learning for steganalysis via convolutional neural networks. In IS&T/SPIE Electronic Imaging, pp. 94090J–94090J. International Society for Optics and Photonics, 2015a.
[7] Haichao Shi, Jing Dong, Wei Wang, Yinlong Qian, Xiaoyu Zhang, SSGAN: Secure Steganography Based on Generative Adversarial Networks. ArXiv e-prints, 2018.
[8] Deep Learning by Ian Goodfellow, Yoshua Bengio and Aaron Courville.
[9] Xiaoyuan, W.Y.N.K.Y., Information hiding scheme based on generative adversarial network. Journal of Computer Applications, 2018.
[10] Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein GAN. ArXiv e-prints, January 2017.
[11] Chintala, Soumith. How to Train a GAN? Tips and tricks to make GANs work. https://github.com/soumith/ganhacks. 2016.
[12] Abadi, Mart´ın, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S, Davis, Andy, Dean, Jeffrey, Devin, Matthieu, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467,2016a.