Spatial-Domain Image Steganalysis using Deep Learning Techniques

Thesis submitted to the
Indian Institute of Technology Guwahati
for the award of the degree

of

Doctor of Philosophy
in
Computer Science and Engineering

Submitted by
Brijesh Singh

Under the guidance of
Dr. Arijit Sur and **Dr. Pinaki Mitra**

Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Feb, 2021

Abstract

Steganography is an art of covert communication where a message is hidden in some natural-looking objects such that no one can even suspect the ongoing communication. It is an ancient practice and is probably originated in Greek mythology. However, presently, the secret message is communicated using a cover of digital media like images, video, text, etc. On the other hand, *steganalysis* is a science for detecting such hidden communication. In this dissertation, we have proposed few steganalysis algorithms which can detect some very recent embedding schemes using the deep learning models.

The conventional image steganalysis methods generally have three primary steps: preprocessing, feature extraction, and classification. It is evident from the literature that features from the steganographic noise domain are relatively more effective for steganalytic detection. Accordingly, few recent steganalysis schemes extract steganographic noise using high pass filtering. They assume that embedding noise generally presents in the image's high-frequency components, but this assumption may not always be valid. Besides, very recently, deep learning-based image steganalysis schemes have become popular for their increasing detection performance. It is also observed that a proper balance between the width and depth of a deeper architecture may improve the detection performance. These observations mainly motivate us to propose few steganalysis algorithms which are primarily based on deep neural models. The four major contributions of this dissertation are as follows.

As we stated earlier, steganalytic classifiers generally work well on the features extracted from the noise domain. It is observed that

steganographic noise may not always reside in the high-frequency zone of the cover image. For example, a recent embedding scheme called *Clustering Modification Directions* (CMD) follows a typical distribution as it tries to embed in neighboring areas with similar embedding directions. In the first part of our first contributory chapter, we propose a steganalytic approach to break the CMD embedding scheme. Most of the recent deep learning-based methods use fixed kernel-based high pass filters to extract steganographic noise, which may not always be very accurate, especially when embedding noise spreads in the low-frequency zone. In the second part of the first chapter, we have introduced a neural network-based dynamic filter kernel. We show that such a denoising kernel can extract steganographic noise more efficiently than fixed kernel-based filters. Subsequently, we proposed a deep classifier trained on the noise residual obtained through our proposed denoising kernel and shown that proposed steganalysis outperforms recent state-of-the-art schemes.

Intuitively, steganographic noise can be detected better if features are learned from different contextual representations. For example, suppose steganographic noise spreads in a relatively homogeneous region. In that case, it can be better tracked with the higher scale space, while noise in the highly textured zone is detected well with lower scale space. In the second contributory chapter, two *Convolutional Neural Network* (CNN)-based methods are proposed, which exploit different contextual representations of the stego image for tracing the embedding more precisely.

The first method uses various sized filters to capture useful steganalytic features using densely connected blocks. The proposed model has no fully connected network, enabling testing any size of images regardless of the image size used for training. It is experimentally shown that the proposed scheme outperformed the existing methods. In the second method, a set of thirty denoising kernels are learned to compute the noise residual. A multi-contextual detection model with

a self-attention mechanism is proposed. The model is trained on the noise residual for accurate detection and is able to outperform the state-of-the-art detectors.

In the recent literature on the deep-learning, it is observed that a balance between the width and depth of a deep model may increase the detection performance effectively. In this line of thought, we have used the concept of Fractal Net as a steganalytic detector in the third contributory chapter. Experimentally, it is found that steganalytic detection of the network increases if the width of the network is increased in a particular proportion to the depth. The proposed deep network is constructed by repeating a basic fractal block so that a balance between the depth and width is maintained. A comprehensive set of experiments reveals that the proposed model outperformed the state-of-the-art methods.

In the final contributory chapter, we have proposed a steganographic embedding model as a data hiding application where we have used a *Generative Adversarial Network* (GAN)-based embedding model to hide an image within another image. The proposed method ensures visual quality, statistical un-detectability, and noise-free extraction by incorporating the perceptual loss function and adversarial training. The proposed framework is tested on various datasets, and results have shown notable improvement ($\sim$ 1 dB) over related existing methods.

Finally, the thesis is concluded by summarizing the significant contributions and proposing some relevant future research directions.

Declaration

I certify that:

a. The work contained in this thesis is original and has been done by me under the guidance of my supervisor.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. I have followed the guidelines provided by the Institute in preparing the thesis.

d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Brijesh Singh

Copyright

Signature of Author.....

Brijesh Singh

Certificate

This is to certify that this thesis entitled **"Spatial-Domain Image Steganalysis using Deep Learning Techniques"** being submitted by **Brijesh Singh**, to Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, for partial fulfillment of the award of the degree of Doctor of Philosophy, is a bonafide work carried out by him under our supervision and guidance. The thesis, in our opinion, is worthy of consideration for award of the degree of Doctor of Philosophy in accordance with the regulation of the institute. To the best of our knowledge, it has not been submitted elsewhere for the award of the degree.

.....

Dr. Arijit Sur
Associate Professor
Department of Computer Science and Engineering
IIT Guwahati

.....

Dr. Pinaki Mitra
Associate Professor
Department of Computer Science and Engineering
IIT Guwahati

Dedicated to

My family & friends

Whose blessing, love and inspiration paved my path of success

Acknowledgments

A great many people have contributed to the production of this dissertation. I owe my gratitude to all those people who have made this possible.

I wish to express my deepest gratitude to my supervisors, Dr. Arijit Sur and Dr. Pinaki Mitra for their valuable guidance, inspiration, and advice. I feel very privileged to have had the opportunity to learn from and work with them. Their constant guidance and support not only paved the way for my development as a research scientist also changed my personality, ability, and nature in many ways. I have been fortunate to have such advisors who gave me the freedom to explore independently and, at the same time, the guidance to recover when my steps faltered. Besides my advisors, I would like to thank the rest of my thesis committee: Prof. Prabin Kumar Bora, Prof. Pardip Kumar Das, and Dr. Ashish Anand for their insightful comments and encouragement. Their comments and suggestions helped me to widen my research from various perspectives.

I also like to express my heartful gratitude to the director, the deans and other management of IIT Guwahati, whose collective efforts have made this institute a place for world-class studies and education. I am thankful to all faculty and staff of the Dept. of Computer Science and Engineering for extending their co-operation in terms of technical and official support for the successful completion of my research work. I want to thank Dr. Arnab Sarkar for supporting and motivating me to overcome the problems either in work or otherwise.

I would like to extend my thanks to Mr. Bhriguraj Borah and Mr. Nanu Alan Kachari for their unconditional technical support.

I am thankful to my friends Dr. Triloki Pant, Sriniwas Pandey, Abhijit Das, Subrata Nandi, Arunangshu Pal, Sathisha B., Basina Deepak Raj, Neelakshi, and Anasua for supporting and motivating me to overcome any problems either at work and otherwise. The countless discussions, sharing of ideas have improved our research.

I am also thankful to all my seniors, friends, and juniors especially, Sibaji Gaj (Gaj da), Satish Kumar (Satish bhai), Shuvendu Rana, Shishendu da, Prasen Kumar Sharma, Anirban, and many others for their unconditional help and support. You made my life at IIT Guwahati memorable.

I am also grateful to all my well-wishers, friends, seniors, and juniors especially Lipika Rekha Sur, Dr. R. Ramakishore, Abhishek Mehta, Akash bhai, Shashank, Priyankar, Rupal, Mohit, Jainam, and many others for their unconditional help and support.

Most importantly, I want to thank my parents, my younger brother Anitesh, my elder borther, my wife, and my little girl Anya, for being a constant source of love, concern, support, and strength all these years.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Contents

List of Figures

LIST OF FIGURES

List of Algorithms

LIST OF ALGORITHMS

List of Tables

List of Acronyms

AUC *Area Under Curve*

bpp *bits per pixel*

BN *Batch Normalization*

CMD *Clustering Modification Directions*

CNN *Convolutional Neural Network*

CPU *Central Processing Unit*

CRMQ1 *Color Rich Model*

DCT *Discrete Cosine Transform*

DFT *Discrete Fourier Transform*

DWT *Discrete Wavelet Transform*

EC *Ensemble Classifier*

FA *False Alarm*

FLD *Fisher Linear Discriminant*

FPR *False Positive Rate*

FLD *Fischer Linear Discriminant*

GAN *Generative Adversarial Network*

GPU *Graphical Processing Unit*

HILL *HIgh-Low-Low*

HUGO *Highly Undetectable steGO*

HVS *Human Visual System*

ILSVRC *Imagenet Large Scale Visual Recognition Challenge*

JPEG *Joint Photographic Expert Group*

JRM *JPEG Rich Model*

LSB *Least Significant Bit*

MD *Missed Detection*

MiPOD *Minimizing the Power of Optimal Detector*

MSB *Most Significant Bit*

MSE *Mean Squared Error*

MS-SSIM *Multi-scale Structural Similarity Index*

NLP *Natural Language Processing*

PRNG *Pseudo-Random Number Generator*

PSNR *Peak Signal to Noise Ratio*

ROC *Receiver Operating Characteristics*

SCAE *Stacked Convolutional Auto-Encoder*

SPAM *Subtractive Pixel Adjacency Matrix*

SRM *Spatial Rich Model*

SSIM *Structural Similarity Index*

SSR *Selective-Signal-Removal*

STC *Syndrome-Trellis Codes*

S-UNIWARD *Spatial UNIversal WAvelet Relative Distortion*

SVM *Support Vector Machine*

TLU *Truncated Linear Unit*

TPR *True Positive Rate*

UQI *Universal-Image-Quality Index*

VIF *Visual Information Fidelity*

WAUC *Weighted Area Under Curve*

WOW *Wavelet Obtained Weights*

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

List of Symbols

μ	The mean
σ	The standard deviation
P_E	The minimum detection error probability
B_{cor}	Block correlation
$\mathbb{BR}$	The subset of $\mathbb{B}$ sustained after re-compression
$\mathfrak{C}$	The minimum Lagrangian cost for PB partitioning
D	Distortion function
ρ_{ij}	The cost of modifying pixel (i, j)
R_{ij}	Noise residual at pixel (i, j)
I_B	Image with block sizes $B \times B$
Var	Variance
ϕ_{DN}	Denoiser subnetwork
$conv$	Convolutional layer
$\mathcal{L}$	Loss function
$\mathbf{T_c}$	Thresholded version of cover image

ϕ_{Eb}	Embedder network
ψ_{Eb}	Discriminator of embedder network
ϕ_{Ex}	Extractor network
ψ_{Ex}	Discriminator of extractor network

Chapter 1

Introduction

The word *steganography* is originated from the Greek word "*stegano-graphia*," which means "*covert-writing*." Steganography is an art of covert communication where a secret message is hidden in an innocent-looking cover medium in such a way that nobody except sender and receiver can even suspect the secret communication.

On the other hand, *steganalysis* is a science to detect the presence of a secret message in an innocuous-looking stego (cover + secret message) media.

Steganography has been studied for centuries. The first steganography technique was reported in the fifth century BC by *Herodotus*, when people used to write secret messages on the wooden table covered with wax over it, the messages tattooed on the slaves shaved head and sent when hair regrew [22, 23]. In 1983, *Gustavus Simmons* published the first paper on steganography which illustrates steganography as the so-called *prisoners' problem* [24].

The prisoners' problem: Alice and Bob are two prisoners locked up in separate cells and wish to create an escape plan. Their communication is allowed by means of sending messages through a trusted courier, as long as they do not discuss escape plans. The warden eve inspects all the communication taking place through the courier. If Eve finds any sign of conspiracy, she will hinder the escape plan by putting both of them in solitary confinement. Alice and Bob

succeed if the communication leads to planning their escape without raising Eve's suspicion.

In the early 1990s, with advancements in storage technology and seamless communication, digital files became the prime choice of steganographers to hide secret messages within the digital medium.

1.1 Steganography system

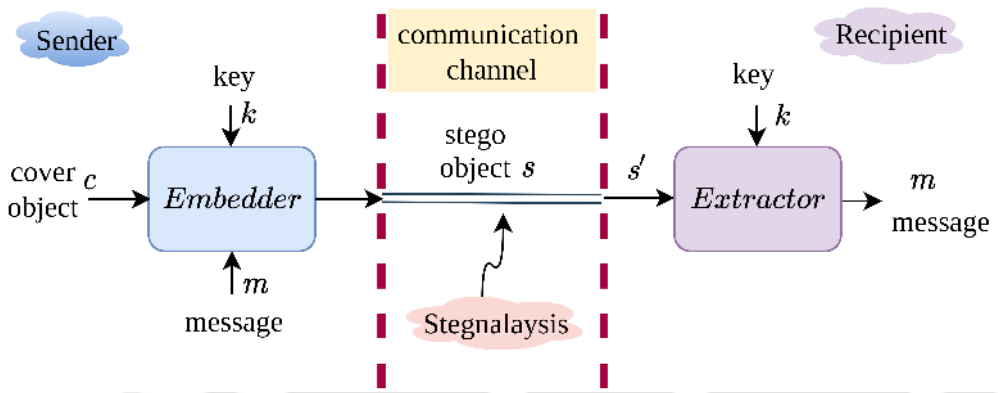


Figure 1.1: *A typical steganograpy system.*

A typical steganography system is shown in Figure 1.1. In general, a steganography system consists of two components - *embedding* (Em) and *extraction* (Ex) algorithms. The embedding algorithm takes three inputs - the cover object (c) where to embed the message, the secret message (m) to be communicated, and the secret key (k). The output of the steganographic embedding is known as the stego object (s). The extraction algorithm takes two inputs - the stego object (s) and the secret key (k). The extraction algorithm extracts the hidden message (m) from the stego object. The mathematical representation of the embedding and extraction process is given by eq. (1.1).

$$\begin{aligned} Em &: c \otimes m \otimes k \rightarrow s \\ Ex &: s \otimes k \rightarrow m \end{aligned} \tag{1.1}$$

1.2 Steganography in Digital Images

Digital images are one of the most popular sources of cover objects due to their frequent usage over the Internet. Till 2017, 46% of total tools available for steganography over the Internet hide messages in digital images [4]. Image steganography is the process of hiding a secret message within an image (known as the *cover image*). The cover image with a hidden message is called a *stego image*. The primary goal of image steganography is to hide a secret message within a cover image such that the *Human Visual System* (HVS) can not perceive the change due to the embedding.

1.2.1 Characteristics of Steganography

The three key characteristics of steganography methods are *imperceptibility*, *capacity*, and *robustness* [25].

- **Imperceptibility:** It means both visual and statistical undetectability. For visual undetectability, there should not be any visual artifacts for embedding with respect to the HVS. For statistical undetectability, we have to ensure that the change in the natural image statistics due to embedding should be low so that any statistical detection tool can't distinguish between the cover and stego image.

- **Capacity:** The embedding capacity (also called payload) is the amount of information that can be hidden in the cover image without degrading its quality. In image steganography, the payload is usually measured in terms of *bits per pixel* (bpp).

- **Robustness:** Robustness refers to the strength of the object to resist steganalytic attacks. In other words, it denotes the degree of difficulty to destroy the hidden information without destroying the cover image. For example, the extraction process should obtain the hidden message even if

the stego image has undergone various image processing operations such as rotation, cropping, etc.

1.2.2 Types of Steganography

Based on the embedding domain, the image steganography methods can be broadly divided into two types - *spatial-domain* steganography and *transformed-domain* steganography.

Spatial-domain steganography: In the spatial domain, images are represented as pixel space. This type of steganography modifies the individual pixel values of the cover image to hide the secret information in the cover image. The spatial-domain steganography allows the steganographer to hide relatively large messages. LSB embedding is an example of spatial-domain steganography.

Transform-domain steganography: In this scheme, the cover image is first transformed to a transform space (e.g., DCT), then the secrete message is embedded to transform coefficients. The transform-domain methods offer high security, strong imperceptibility, and robustness.Examples of transform-domain schemes are *Discrete Cosine Transform* (DCT), *Discrete Wavelet Transform* (DWT), *Discrete Fourier Transform* (DFT), etc.

1.3 Steganalysis in Digital Images

The steganalysis in digital images refers to analyzing innocuous-looking images to detect the trace of hidden information. Steganalysis can be thought of as the action performed by the warden (*Eve*) in the *prisoners' problem*. Specifically, the warden can be categorized into two types - *active* and *passive* warden.

The active warden technique deliberately checks and modifies the object in transmission to destroy the hidden information, even if it is clean. In the passive warden technique, the object in the transmission is examined to determine

whether it contains a piece of covert information. Active and passive wardens are shown in Figures 1.2 and 1.3, respectively.

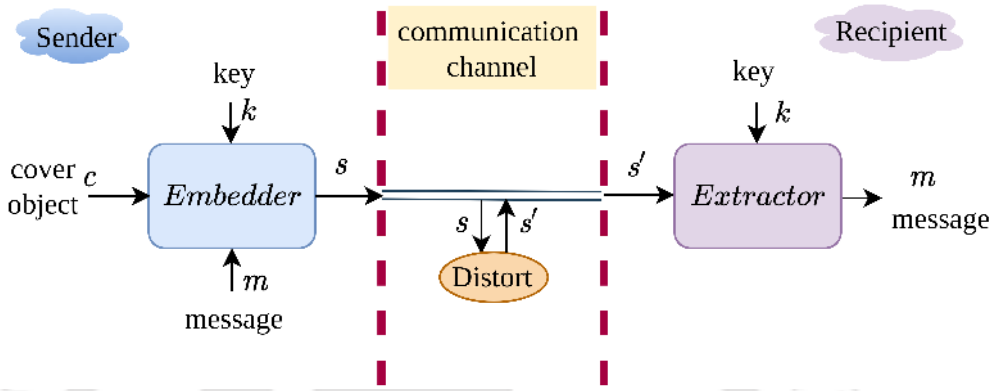


Figure 1.2: *The Active warden technique. The object is distorted before sending to the receiver.*

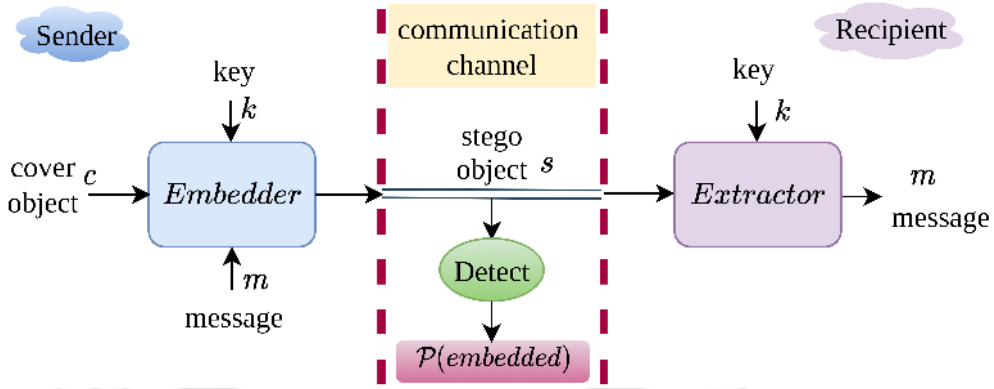


Figure 1.3: *The Passive warden technique. The hidden infomation is detected when stego object is transmitted.*

1.3.1 Types of Steganalysis

Based on the steganographic method's prior knowledge, steganalysis methods can be categorized into two types- *targeted/specific* and *blind/universal*.

Targeted steganalysis: In the targeted method, the steganalyst knows the steganographic embedding algorithm used for hiding the message. In general,

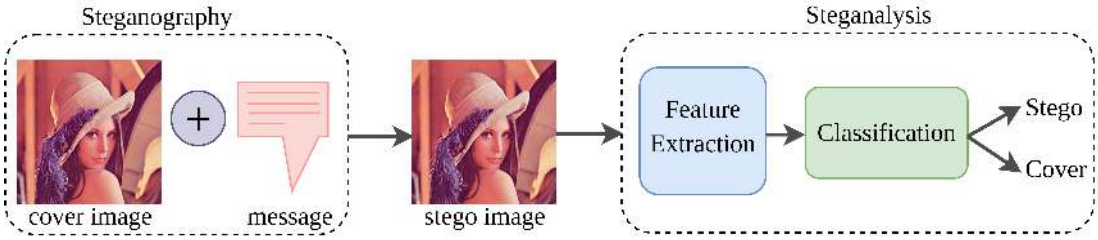


Figure 1.4: *A typical steganalysis process.*

the targeted methods lookup for specific distortions.

Blind steganalysis: Blind methods do not use any prior knowledge about the embedding algorithm used. These methods are useful when the source of the object under analysis is unknown. In practice, blind methods are more generic tools for detection but may not work well for every steganographic scheme.

1.3.2 Steganalysis Process

Every steganalysis follows a standard procedure for detecting the trace of hidden information. First, feature extraction is performed to extract all the essential features required for classifying the given image as stego or cover. Second, based on the extracted features, a classification model decides whether the given image is stego or cover. A typical steganalysis process is shown in Figure 1.4.

1.3.3 Applications

Image steganography covers a broad area of applications for security and privacy:

- **Medical Imaging:** Steganography is used to embed the patient's record into the image, reduce transmission time, and provide protection of information [26].
- **Military Agencies:** Steganography is known to be used by military agencies to provide anonymous communication over the Internet [27].

- **Intelligence Agencies:** Intelligence agencies use steganography for the safe transmission of confidential data [28].

- **Smart Identity cards:** In smart identity cards, the user's personal details are embedded in the photograph itself [27].

- **Document Authentication:** Steganography allows storing additional information such as owner identity and the document's subject in the document itself for verification [27].

- **Steganographic File systems:** Steganographic file system safely masks given files in a file system such that an eavesdropper will not be able to guess their presence without the corresponding access keys, even though the eavesdropper is thoroughly familiar with the file system and has obtained complete access to it [29].

1.4 Literature Survey

This section presents a literature survey of the works related to steganography and steganalysis in the spatial domain.

1.4.1 Steganography Schemes

Steganography systems have evolved over the years to improve security of the information exchange. In the spatial-domain schemes, the covert message is embedded by replacing redundant pixel bits of the image. The steganographic schemes can be roughly categorized as the *Prior arts: Least Significant Bit* (LSB) embedding/replacement etc., *Recent Methods:* Cost function-based embedding, adaptive embedding, etc., and *Deep-Learning based techniques:* These methods utilize CNN and GAN to design steganographic frameworks.

1.4.1.1 Least Significant Bit (LSB) steganography

Least Significant Bit (LSB) steganography takes advantage of the fact that a small perturbation to a pixel of an image is not perceivable by the HVS. In this method, the least significant bit of the selected pixel is replaced with the message bit. The selection of the pixels for replacement is either successive or pseudo-random. In successive selection, the consecutive pixels of the cover image are replaced with the message bits. In pseudo-random selection, a key is used as a seed for a *Pseudo-Random Number Generator* (PRNG). The PRNG generates a random number that specifies a pixel to be modified. Nonetheless, the stego image embedded by the LSB scheme is susceptible to noise and can be easily destroyed by any modification to the stego image. A variety of LSB steganography can be found in [23, 30–33].

1.4.1.2 Content-adaptive Steganography

The idea of content-adaptive methods is evolved in recent times. The objective of these methods is to embed a secret message by minimizing a heuristically defined distortion function. The development of these methods is usually driven by the detection performance of steganalysis [34]. In these steganography algorithms, the distortion caused by the embedding modifications relies on the local image content, which results in high embedding modification in the texture regions and low in the smooth regions of the image. Consequently, the name is *content-adaptive* steganography. An example of the distortion function used by these methods is given in eq. (1.2).

$$D(X,Y) = \sum_{i=1}^{M}\sum_{j=1}^{N} \rho_{ij}|X_{ij} - Y_{ij}|, \tag{1.2}$$

where ρ_{ij} is the costs of modifying pixel X_{ij} to Y_{ij}, and X and Y, represent the cover image and the corresponding stego images, respectively, each with size $M \times N$. The embedding of the message to the cover image is done by minimizing

the distortion function, $D(X, Y)$, using *Syndrome-Trellis Codes* (STC) [35]. Some of the recent content-adaptive steganography methods are listed below.

HUGO: *Highly Undetectable steGO* (HUGO) [6] is a secure steganography method that uses high-dimensional image models. HUGO used a steganalysis method called SPAM [13] to compute the high-dimensional features. The distortion function is a weighted sum of differences of SPAM features of cover and corresponding stego images. This design of HUGO enables the embedding to take place primarily in the textured area and edges.

WOW: *Wavelet Obtained Weights* (WOW) [5] utilizes a collection of directional high-pass filters to compute the directional residuals and determine the content around each pixel in different directions. This method measures and aggregates the embedding impact for each directional residual, thereby forcing the high distortion to the predictable areas in at least one direction (clean edges and smooth regions) and low distortion to the unpredictable regions in every direction (texture regions). The WOW algorithm is highly adaptive and more secure than the HUGO.

S-UNIWARD: *Spatial UNIversal WAvelet Relative Distortion* (S-UNIWARD) [15] used a so-called universal distortion function that does not depend on the embedding domain. The distortion is computed as a sum of relative changes of wavelet coefficients. The S-UNIWARD utilizes the wavelet filter bank of the WOW steganography. This method is designed for spatial as well as JPEG domains.

HILL: *HIgh-Low-Low* (HILL) [18] steganography uses the smoothed residual to model the embedding distortion, unlike WOW and S-UNIWARD, where weighted filter residual difference is used. The HILL algorithm uses a high-pass filter (Ker-Bohme filter eq. (1.5)) followed by low-pass filters to identify the unpredictable

regions. The HILL steganography showed better performance against SRM steganalysis as compared to HUGO, WOW, and S-UNIWARD.

CMD steganography: *Clustering Modification Directions* (CMD) steganography [36] works as a wrapper for steganography schemes by dynamically varying the embedding costs to attain a lower detection rate. The dynamic cost function allows the algorithm to take advantage of mutual embeddings and clustering of similar distortions. This clustering ensures that the embeddings in neighboring pixels are of similar nature (± 1), which increases the steganographic security. The CMD approach is very general and robust and can be applied to any existing additive steganographic algorithm, such as WOW [5], S-UNIWARD [15], and HILL [18]. When the wrapping of CMD is applied over any of the aforementioned steganography schemes, the steganalytic performance of SRM [14] degrades by $\sim 5\%$.

1.4.1.3 Steganography Using GAN

With the recent evolution of deep-learning, researchers have explored various aspects of deep-learning; specifically, *Generative Adversarial Network* (GAN) [37], to solve steganographic problems, such as designing embedding distortion functions, generating secure cover images for embedding, hiding the message in images, etc.

Embedding Distortion Learning Using GAN: Tang *et al.* proposed the ASDL-GAN [38] for automatic learning of steganographic distortion function. ASDL-GAN uses a generator network with 25 layers of the CNN model, which predicts the embedding change probabilities for every cover image pixel. The predicted probabilities are used as embedding distortion. The XuNet [39] is used as the discriminator of ASDL-GAN to ensure the undetectability against state-of-the-art detectors. However, the performance of the ASDL-GAN could not compete with that of the handcrafted steganography scheme S-UNIWARD. In

a similar direction, Yang *et al.* proposed a UT-GAN [40] to predict embedding change probability from a given cover image. The UT-GAN uses U-Net [41] architecture as the generator and modified XuNet architecture as the discriminator. The authors showed that the UT-GAN outperformed the steganography schemes ASDL-GAN, S-UNIWARD, HILL, and MiPOD.

Secure Cover Generation Using GAN: Volkhonskiy *et al.* [42] devised a GAN-based model called *SGAN* to generate container/cover images from a noise distribution. The generated images are embedded using ± 1 embedding to obtain corresponding stego images. Haichao *et al.* [43] proposed a model similar to SGAN [42], namely *SSGAN*. The SSGAN model uses Wasserstein distance as a loss function [44] for training. The steganalyzer used in SSGAN is well known (one of the early CNN-based steganalyzers) GNCNN [2]. The authors showed that the SSGAN generated cover is more secured than SGAN. The cover images produced by [42, 43] are more secure than the ordinary cover images. However, the generated images can be easily suspected as they look unrealistic. Therefore, these images may not be used as the cover image in practice.

Information Hiding Within an Image Using GAN: Hayes and Danezis [45] proposed a generative adversarial model comprised of a generator, a steganalyzer, and an extractor. The generator hides an n-bit binary message within a cover image, and the extractor retrieves the secret message from the stego image. Baluja [20] used an adversarial training approach to hide one image in another image of the same size. In [21], Baluja extended this approach to hide multiple images in an image. This approach only uses MSE loss for both embedder and extractor and has not used any discriminator to impose undetectability. Zhang *et al.* proposed ISGAN [46] to hide and recover a grayscale image within the Y-channel of an image using GAN. ISGAN used SSIM and MS-SSIM loss for training to improve the imperceptibility. Zhu *et al.* [47] used a GAN-based network to embed and extract a bit string encoded message in the image. Zhang *et*

al. [48] introduced SteganoGAN to hide/extract binary data to/from images. The methods [47, 48] have not used extractor loss in training, resulting in improper embedding and might show visual artifacts in generated stego images.

1.4.2 Steganalysis Schemes

Steganalysis is the complementary process of steganography, where the goal of steganalysis is to detect the traces of hidden messages in the innocuous-looking stego object. Steganalysis involves feature extraction followed by classification between the cover and stego features. Based on the feature representation, the steganalysis methods can be broadly categorized into two types, namely, *Handcrafted* and *Deep* feature-based.

1.4.2.1 Handcrafted Feature-based Steganalysis

Handcrafted feature-based methods exploit the fact that the recent content-adaptive steganography schemes perform high embedding to the texture areas and low to the smooth regions of the image. Consequently, these methods employ various handcrafted high-pass filters to extract the features from the texture regions. These features are then used to train machine learning-based classifiers to detect steganographic embedding traces in the image.

Pevny *et al.* devise *Subtractive Pixel Adjacency Matrix* (SPAM) [13] for steganalysis in digital images. SPAM method identified the deviations between neighboring pixels, which are due to the steganographic perturbation. This model applied a high-pass filter to suppress the image component and expose the stego noise. A higher-order Markov chain [49] is used to model the dependencies between the neighboring pixels of the noise residual. The resulting transition probability matrix of the Markov chain is used as a feature to train a soft-margin *Support Vector Machine* (SVM) [50] with a *Gaussian* kernel for classifying the image as cover or stego.

Fridrich & Kodovsky devised *Spatial Rich Model* (SRM) [14] for steganalysis. The objective of SRM is to capture various dependencies among the neighboring

pixels, which enables the model to detect embedding changes in the texture regions. SRM uses various small submodels to capture these dependencies. Each submodel is created from noise residual, R_{ij}, computed using high-pass filters as follows:

$$R_{ij} = \hat{X}_{ij}(Nij) - c.X_{ij}, \tag{1.3}$$

where c denotes the order of residual, N_{ij} is the neighbors of pixel X_{ij}, $X_{ij} \neq N_{ij}$, and $\hat{X}_{ij}(.)$ is a predictor of $c.X_{ij}$ defined over N_{ij}. The set $\{X_{ij} + N_{ij}\}$ is known as *residual's support*. The residuals are quantized and truncated as follows:

$$R_{ij} \leftarrow round\left(trun_T\left(\frac{R_{ij}}{q}\right)\right), \tag{1.4}$$

where $trun_T(x) = x$ if $x \in [-T, T]$, and $q > 0$ denote the quantization parameter. The truncation step limits the dynamic range of residuals which results in the reduced dimensionality of co-occurrence matrices [14]. The quantization causes the residual more sensitive to the embedding modification in spatial discontinuities such as edges and textures. The co-occurrence matrix (of order $D = 4$) is computed from the quantized residual (with $T = 2$, and $q \in \{1, 1.5, 2\}$) of entire image. This co-occurrence matrix is used as an SRM feature vector to train the *Ensemble Classifier* (EC) [17]. The EC used with SRM features consists of L *Fischer Linear Discriminant* (FLD)s as base learners due to their simplicity and fast training. The optimal value of L is automatically learned during ensemble training. The decisions of base learners are aggregated by majority voting. SRM is one of the successful steganalyzers. Other different variations of SRM, such as CRMQ1 [51], JRM [52], etc., were subsequently reported in the literature. Though handcrafted feature-based schemes are known to detect state-of-the-art steganalysis schemes successfully, their detection performance relies heavily on handcrafted filters' design.

1.4.2.2 Deep Feature-based Steganalysis

In recent times, deep learning methods, specifically, *Convolutional Neural Networks* (CNNs), have shown tremendous success for many computer vision applica-

tions, such as object detection [53], object tracking [54], image classification [55], medical imaging [56], etc. This success drove the researchers to explore CNN for steganalytic classification.

Tan & Li [57] proposed a deep learning architecture based on a nine-layered *Stacked Convolutional Auto-Encoder* (SCAE) to simulate the SRM for steganalytic detection. For example, the first layer consists of forty kernels, initialized using a high-pass filter (KV filter) given in eq. (1.5). The method used max-pooling in the subsequent layers for dimensionality reduction. Finally, a fully-connected layer, followed by a softmax layer, is used for classification. The detection performance of the method was inferior to that of the handcrafted feature-based SRM. This method is the first work that used deep learning for steganalysis. Therefore, various design considerations are not taken into account, such as avoiding the use of max-pooling.

$$KV = \frac{1}{12}\begin{pmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{pmatrix} \tag{1.5}$$

Qian *et al.* [2] proposed a shallow CNN architecture with a Gaussian activation function, called GNCNN, to model the cover signal as 0 and stego signal as $+1$ or -1. GNCNN used the KV filter in preprocessing step to increase the signal-to-noise ratio (SNR), thereby suppressing image content and exposing noise content in preprocessing step to allow better extraction of stego noise. GNCNN was the first CNN model, which attained steganalytic detection close to the SRM. However, the shallower architecture of GNCNN might not have good discriminative power for steganalytic detection.

Xu *et al.* [39] devised a structural design of CNN (XuNet), which includes absolute layers (ABS) for better modeling of the stego noise (negative as well as positive embeddings) and *Batch Normalization* (BN) [58] to evade the CNN model from falling in the local minima when training. XuNet also used the

KV filter in preprocessing step of the network. The structural design of XuNet enabled the model to attain performance competitive to the SRM.

Qian *et al.* [59] presented a transfer learning paradigm, where the model (GNCNN) is trained using images embedded with a high payload. The trained weights are then finetuned using lower payloads. The steganalytic detection performance of the model was better than that of SRM on WOW steganography with low embedding rates.

Ye *et al.* [60] introduced a CNN architecture (YeNet) where layers were initialized with the thirty high-pass filters of SRM and TLU activation to limit the residual within a confined range. They also used data-augmentation [61] as regularization to improve the training of the steganalyzer.

Li *et al.* [16] proposed a steganalyzer with a diverse-activation-module called *ReSTNet*. ReSTNet was inspired by the observation that increasing the width of the network boosts the steganalytic detection performance. The ReSTNet architecture is formed by combining three pre-trained CNNs. Each CNN preprocessed the input image using a different type of filter - Gabor [62], SRM linear, and non-linear filters and also used distinct activation functions - ReLU, Sigmoid, and TanH. This diverse design of the model performed better than the existing approaches.

Yedroudj *et al.* [63] proposed a steganalyzer by fusing state-of-the-art detectors. Yedroudj-Net used thirty filters from SRM in the preprocessing step, similar to YeNet. After preprocessing, the model uses five convolution layers for feature learning, followed by a fully-connected network.

Boroumand *et al.* [4] proposed an end-to-end deeper CNN model that utilizes skip connections [64] for steganalytic detection. The first seven layers of the model compute the noise residuals, and the rest five layers performed the steganalytic detection using the computed noise residuals. Unlike other existing methods, this is the first method, which does not use any fixed high-pass filter for preprocessing.

1.5 Motivation and Objective

Steganography is a well-explored area where plenty of embedding schemes and counter detection (steganalysis) schemes are reported in the last two decades. With the emergence of the deep-learning paradigm, embedding and corresponding detection methods are a bit modified. Thus, it can be said that the challenges for both embedding and detection procedures are a bit changed. We have observed the following issues in the recent scientific literature:

- Nowadays, embedding is not restricted to high-frequency zones of the image; rather, it also spreads over low-frequency zones (e.g., CMD embedding [36]). Consequently, older detection techniques which assume mostly high-frequency-zone-based embedding, may not work well for these methods. In this line of thought, extraction of the embedding noise may be improved by using a dynamic (i.e., learnable) filter kernel.

- In recent literature, most of the existing steganalysis methods considered the fixed context sizes of filters in subsequent layers while designing the CNN-based steganalyzers. However, it is observed that multi-contextual features collected from different scale-space of the image may help better model the embedding noise.

- Most of the existing steganalysis approaches used a deeper network architecture to learn the steganographic noise. However, a recent study [65] showed that, for an end-to-end deep-learning-based detection model, a suitable balance between width and depth of the network performs notably well and may be used as a detection network.

- Most of the recent GAN-based steganography schemes designed to hide an image within another image suffer from visual artifacts and lack undetectability due to the limitations, such as the absence of suitable loss function, feedback from the extractor, and a better discriminator to impose undetectability.

Motivated by the above observation and limitation of the existing works, the main objectives of this thesis are as follows:

- Understand the embedding procedure of CMD [36] steganography and devise a steganalysis approach to detect the CMD steganography.
- Propose a deep learning-based steganalyzer that can learn the preprocessing filter instead of using a fixed one, thereby improving the steganalytic detection performance.
- Develop a deep learning-based steganalysis model that can learn the sparse characteristics of noise residuals and attain improved detection.
- Devise an end-to-end deep learning-based steganalyzer whose design is focused on depth as well as the width for better learning of steganographic features.
- Finally, devise a GAN-based steganography method that can hide an image within another image, which can ensure imperceptibility and undetectability at the same time.

1.6 Contribution of the Thesis

The main contributions of this thesis are as follows.

1.6.1 Breaking CMD Steganography and Kernel Learning for Steganalysis

In the initial part of our first contributory chapter, we propose a steganalytic approach to break the CMD steganography scheme [36]. Empirical evaluation shows that the method effectively identifies the embedding regions and nullifies the effect of CMD steganography. In the second part of this chapter, we have introduced a convolutional neural network-based dynamic filter kernel. We show that such a denoising kernel can extract steganographic noise more precisely than

fixed kernel-based filters. Subsequently, we proposed a deep classifier trained on the noise residual obtained through our proposed denoising kernel and schemes.

1.6.2 Multicontextual Design of CNN for Steganalysis

In the second contributory chapter, two CNN-based methods are proposed, which exploit different contextual representations of the stego image for tracing the embedding more precisely. The first method uses various sized filters to capture useful steganalytic features using densely connected blocks. The proposed model has no fully connected network, enabling testing any size of images regardless of the image size used for training. It is experimentally shown that the proposed scheme outperformed the existing methods. In the second method, a set of thirty denoising kernels are learned to compute the noise residual. A multi-contextual detection model with a self-attention mechanism is proposed. The model is trained on the noise residual for accurate detection and is able to outperform the state-of-the-art detectors.

1.6.3 Steganalysis using Fractal Architecture : The Role of Network Depth and Width in Steganalysis

In the recent literature on the deep-learning, it is observed that a balance between width and depth of a deep model may increase the detection performance effectively [65]. In this line of thought, we have used the concept of Fractal Net [66] as a steganalytic detector in the third contributory chapter. Experimentally, it is found that steganalytic detection of the network increases if the width of the network is increased in a particular proportion to the depth. The proposed deep network is constructed by repeating a basic fractal block so that a balance between the depth and width is maintained. A comprehensive set of experiments reveals that the proposed model outperformed the state-of-the-art methods.

1.6.4 Hiding Image within Image using Conditional GAN: An application of Steganalysis

In the final contributory chapter, we have proposed a steganographic embedding model as a data hiding application where we have used a GAN-based embedding model to hide an image within another image. The proposed method ensures visual quality, statistical un-detectability, and noise-free extraction by incorporating the perceptual loss function and adversarial training. The proposed model also uses a state-of-the-art steganalyzer called XuNet [39] as the discriminator to impose undetectability. The proposed model is tested on various datasets, and results have shown notable improvement (∼1dB) over related existing methods.

1.7 Organization of the Thesis:

This PhD thesis consists of seven chapters. The first chapter consists of a brief introduction to steganography and steganalysis, a brief literature survey, research motivation and objectives, contribution of the thesis. The rest of the thesis is organized as follows,

- Chapter 2 presents the requisite background of the research, such as the preliminary concepts of convolutional neural networks, evaluation metrics, and the datasets used in carrying out the experimentation.
- In Chapter 3, a steganalysis method is presented for the detection of CMD steganography. Further, a deep learning-based steganalysis model is proposed, which replaces the fixed high-pass filter used in the preprocessing stage of steganalyzers with a filter learned by a CNN.
- Chapter 4 proposed a multi-contextual framework for steganalysis. The proposed model uses the different context sizes in each layer of the network to learn sparse noise residuals for steganalytic detection.
- Chapter 5 presents a novel deep learning-based steganalyzer that is inspired

by the Fractal network. The proposed model is designed by considering the balance between the depth and width of the network.

- Chapter 6 devises a conditional generative adversarial model to hide an image within another image. The model preserves imperceptibility as well as undetectability by employing suitable loss functions and a state-of-the-art steganalyzer as the discriminator.

- In Chapter 7, this thesis is concluded along with the future scope of the research directions.

1.8 Summary

In this chapter, a brief introduction of steganography and steganalysis is presented to formulate the scope of research works. First, the concept of steganography and steganalysis is explained. Then, brief literature on steganography and steganalysis is discussed. Based on the shortcomings of the existing literature, the objectives of the research are formulated. Finally, a brief description of the contributions and the organization of the thesis is presented.

Chapter 2

Research Background

In this chapter, a brief overview of some fundamental concepts related to the topics of interest is presented. It includes a brief introduction to *Convolutional Neural Network* (CNN) such as *VGGNet* [67], *ResNet* [64], *DenseNet* [68], and *FractalNet* [66]. The concepts of these CNNs are used to construct the proposed deep learning steganalyzers to detect the stego images. In addition, various evaluation metrics for evaluating the proposed methods and corresponding datasets used for experiments are also discussed in this chapter.

2.1 Convolutional Neural Networks

In the recent deep learning paradigm, *Convolutional Neural Network* (CNN) has shown tremendous success for tasks in various domains such as Computer Vision [55, 67, 69–71], *Natural Language Processing* (NLP) [72–75], Speech processing [76–78], etc. The ability to automatically learn task-specific characteristics allowed CNNs to achieve state-of-the-art performance on different benchmarks effectively. The use of CNNs has been made possible by recent advances in computer hardware and the availability of very large-scale data. In general, a CNN architecture is comprised of the following components: convolutional layers, activation function, pooling layers, and fully connected layers.

The detailed description of each of these components are as follows:

1. **Convolutional Layers:** The parameters of the convolutional layer consist of a collection of learnable filters. Each filter is spatially small but extends across the full depth of the input volume. When an input image is presented to a convolutional layer, each filter moves over the height and width of the input image and obtains a dot product between the inputs in the filter and the input at any position. It generates a two-dimensional activation map representing some similarity measure between the kernel and the corresponding image patch as the filter moves through the input volume. Each value of the activation map corresponds to the response of the filter at each spatial location in the input. These activation maps are further given as input to an activation function.

2. **Activation Function:** The activation function generally put some non-linearity over the convoluted feature maps. Some of the popular activation functions are as follows:

   (a) *Rectified Linear Unit* (ReLU): $f(z) = max(0, z)$

   (b) *Hyperbolic Tangenet* (TanH): $f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

   (c) *Sigmoid*: $f(z) = \frac{1}{1+e^{-z}}$

   (d) Leaky ReLU: $f(z) = \begin{cases} z, & \text{if } z \geq 0 \\ \alpha.z, & \text{otherwise} \end{cases}$,
   where α is a constant. Usually, $\alpha = 0.001$.

   (e) *Parametric ReLU* (PReLU): $f(z) = \begin{cases} z, & \text{if } z \geq 0 \\ \alpha.z, & \text{otherwise} \end{cases}$,
   where α is a hyperparameter learned together with the model parameters.

3. Pooling Layer: The pooling layer gradually reduces the number of parameters and spatial size of the representation and controls overfitting. On each depth slice of the input, the pooling layer works independently and resizes it spatially, using different pooling operations like max pooling, average pooling, etc. Pooling layers are usually applied to the representation

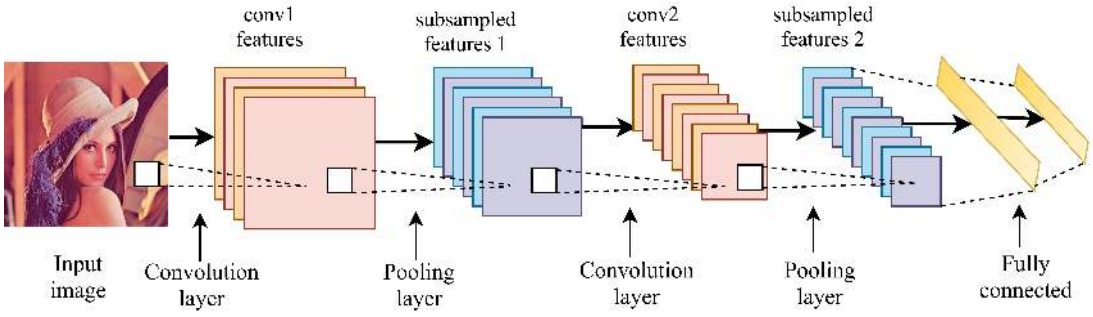


Figure 2.1: *An example of CNN architecutre.*

produced by the activation function. For example, a pooling layer with 2×2 size filters with the stride of 2 downsamples each slice of depth in the input by 2 along both width and height. This operation discards 75% of the activations. The popular types of pooling operations are Max Pooling and Average Pooling.

4. Fully Connected Layers: Like the usual neural networks, neurons in a fully connected layer have full connections to all activation in the previous layer. Therefore, the activations of fully connected layers can be computed using a matrix multiplication followed by a bias offset.

In general, the first three components constitute a single layer of a CNN. For deeper CNN, such layers with different hyper-parameters are placed subsequently, followed by the one or more fully connected layers. An example of CNN is given in Figure 2.1. The initial convolutional layers learn to extract the high-level features such as edges, lines, etc. As layers go deeper, the convolutional layers learn to extract the fine-grained features. These features are further learned by the fully connected layers to accomplish the intended task. In this thesis, the concepts of some of the existing deep CNN models - a) *VGG-16*, b) *ResNet*, c) *DenseNet*, and d) *FractalNet* are utilized to design the efficient steganalyzers.

2.1.1 VGG-16

VGG-16 [67] is one of the popular deep convolutional neural networks proposed by Simonyan and Zisserman for the task of *Imagenet Large Scale Visual Recognition Challenge* (ILSVRC) 2014 [7]. It was the runner-up of the ILSVRC 2014 challenge. The architecture of *VGG-16* is shown in Figure 2.2. An RGB image

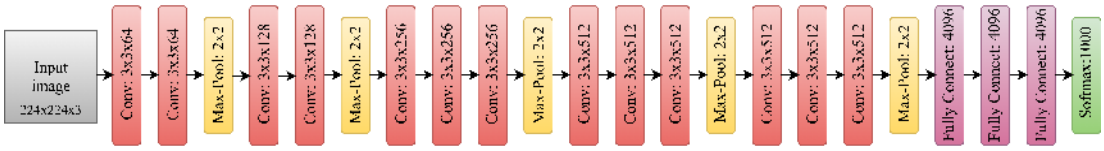


Figure 2.2: *VGG-16 model architecture.*

with dimensions $224 \times 224 \times 3$ is given to the input of the network. All the convolutional layers have a filter of size 3×3. All the layers except the last fully connected use ReLU as the activation function. The stride is fixed to 1 pixel to span each pixel of the image, and padding is kept as 1 to retain the same spatial dimension as the input. Max pooling is applied after a set of convolutional layers with a 2×2 window and stride of 2. *VGG-16* outperformed the performance of *AlexNet* [55] on large-scale image recognition task [7] by attaining 7.3% Top-5 error.

2.1.2 ResNet

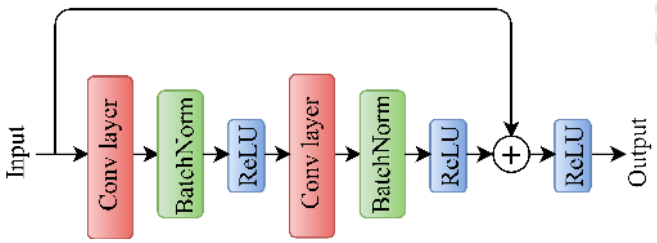


Figure 2.3: *A residual building block.*

The depth of convolutional neural networks has been growing in terms of the number of layers. For example, *VGG-16* was deeper as compared to the *AlexNet*.

Nevertheless, only increasing the network depth may not improve the performance of the network always; instead, it may get degraded [64]. This degradation problem is addressed by He *et al.* through Residual Network (*ResNet*) [64]. The main idea of *ResNet* is "skip connection," as shown in Figure 2.3. The skip connections skip a set of layers, which does not increase the number of parameters, and it simply sums the output activations from the previous layer to the layer ahead. The authors suggested that it is easier to optimize the residual mapping than the original mapping. This configuration of *ResNet* also solved the vanishing gradient problem. *ResNet* was the winner of ILSVRC 2015. *ResNet* showed the best performance compared to the existing models for large-scale image classification tasks by achieving a 3.57% Top-5 error.

2.1.3 FractalNet

FractalNet [66] architecture shown in Figure 2.4 is based on self-similarity and is generated by expanding the basic fractal/block by using the expansion rule shown in Figure 2.5.

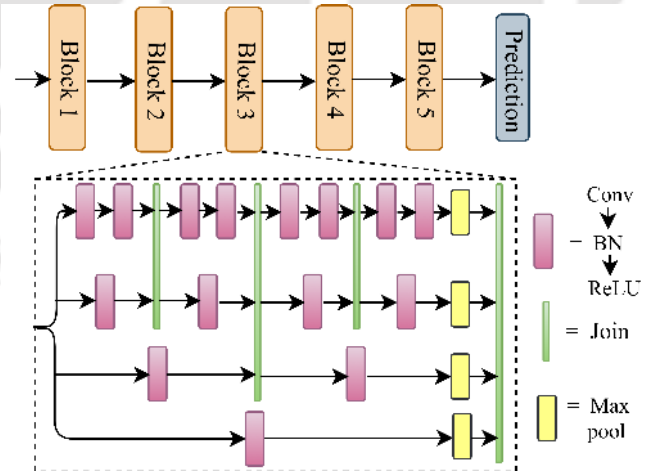


Figure 2.4: *FractalNet architecture*

Formally, let k be the number of intertwined columns or width. The base case $f_1(x)$ contains a single layer of convolution between input and output. i.e.,

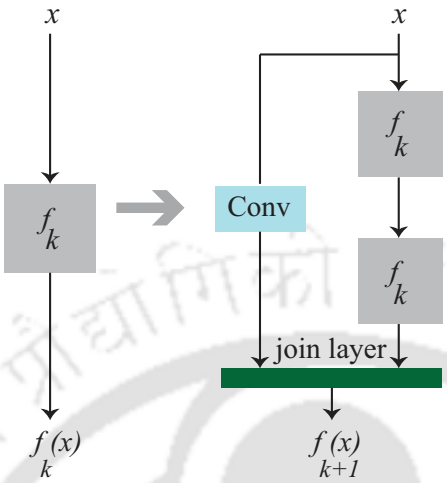


Figure 2.5: *Fractal expansion rule*

$f_1(x) = conv(x)$. Successive fractals can be recursively defined using the following rule:

$$f_{k+1}(x) = [f_k \circ f_k(x)] \oplus [conv(x)],$$

where $\circ$ represents the composition and $\oplus$ denotes the join operation between two different blocks. The value k denotes the width of the network. The depth of the network, which is the longest path between the initial and final layer, can be defined as 2^{k-1}. This organization of *FractalNet* forces the network to vary the depth and width of the network proportionally. The *join layer* combines the features from two or more incoming paths. The features from the incoming connections can be joined using *sum*, *max-out*, *average*, or *concatenate*. In the latter case, the number of channels in the subsequent layers may increase. *FractalNet* also uses drop-path regularization to force each input to the *join* layer to be individually significant. Further, the *FractalNet* is comprised of several fractal blocks connected using pooling layers. *FractalNet* has shown a competitive performance to the ResNet [64] for the image recognition task.

2.1.4 Generative Adversarial Networks (GAN)

Generative Adversarial Network (GAN) [37] is a deep learning framework proposed by Goodfellow *et al.* for estimating generative models using adversarial training. In adversarial training, two models, a generative model (G) and a discriminative model (D), are trained simultaneously. G strives to capture data distribution, and D estimates the probability that the given sample came from training data or is generated by G. To understand the procedure mathematically, let p_g represent the generator's distribution over data x, $p_z(z)$ denotes the input noise variables. $G(z;\theta_g)$ denotes a mapping to data space, where G is a differentiable function with parameters θ_g. Let $D(x;\theta_d)$ be another differentiable function with parameter θ_d, which outputs a single value. $D(x)$ denotes the probability that x has drawn from data rather than p_g. D is trained to maximize the likelihood of correctly classify both training examples and generated samples by G. At the same time, G is trained to (fool the D) minimize $log(1 - D(G(z)))$. G and D play the two-player minimax game using eq. (2.1).

$$\min_G \max_D \mathcal{L}(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\left[logD(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[log(1-D(G(z)))\right] \tag{2.1}$$

This network setting enables the generative model to estimate data distribution so that the discriminator fails to classify between the original and generated data.

2.2 Evaluation Metrics

The Steganography and Steganalysis algorithms can be evaluated using *quantitative* and *visual* metrics.

2.2.1 Quantitative Metrics

A variety of metrics, such as the minimum detection error probability, *Receiver Operating Characteristics* (ROC), *Area Under Curve* (AUC), *Weighted Area Under Curve* (WAUC), are used to show the performance of a steganalyzer quantitatively. These metrics are outlined below.

- **Minimum Detection Error Probability:** A steganalytic detection system is consists of three main components - Input, Steganalytic detector, and Output. (i) The input image may be either cover (natural image) or a stego image (embedded image). (ii) The steganalytic detector is a model that is trained to identify that the given input image is cover or stego. (iii)The output of the steganalyzer is the decision that the given image is cover or stego. When a model is tested to classify a given image I, one of the following two errors may be possible:
  False Alarm (FA): a cover image C (negative) is incorrectly classified as stego image S (positive).
  Missed Detection (MD): a stego image S is incorrectly classified as cover image C. Let P_{FA} and P_{MD} denote the probability of false alarm and missed detection, respectively. The conditional probabilities can be defined as follows:

$$P_{FA} = \mathcal{P}(S \mid I \in C)$$

$$P_{MD} = \mathcal{P}(C \mid I \in S)$$

  Therefore, the total error probability P_E can be defined as:

$$P_E = P_{FA} \,.\, \mathcal{P}(I \in C) + P_{MD} \,.\, \mathcal{P}(I \in S)$$

  Here, $\mathcal{P}(I \in C) = \mathcal{P}(I \in S) = \frac{1}{2}$.

$$P_E = \frac{1}{2}(P_{FA} + P_{MD}) \tag{2.2}$$

  In general, for steganalysis, minimum error probability under equal priors is used for the evaluation of a model. The minimum error probability under equal priors can be written as:

$$P_E = \min_{P_{FA}} \frac{1}{2}(P_{FA} + P_{MD}) \tag{2.3}$$

- **Receiver Operating Characteristics (ROC) Curve:** The ROC curve is a metric used to assess a binary classifier's performance. This curve is

created by plotting the *True Positive Rate* (TPR) against the *False Positive Rate* (FPR) at different threshold values. The area under the ROC curve is called *Area Under Curve* (AUC). An example of a ROC curve is shown in Figure 2.6. The ROC curve is depicted in the "blue" color. The shaded region indicates the AUC. The value of AUC is indicated on the top-left corner of the plot.

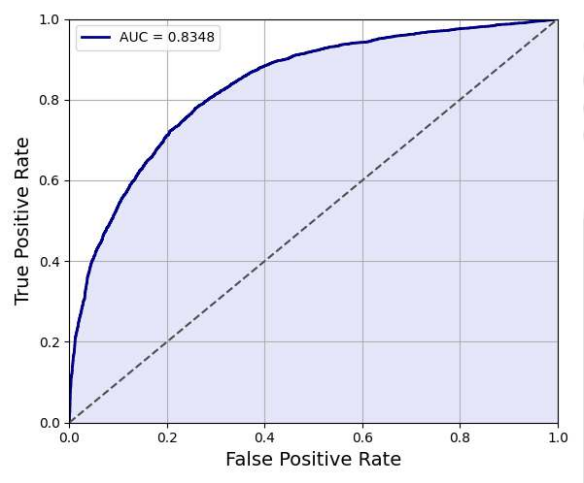


Figure 2.6: *An example of ROC curve.*

- **Weighted Area Under Curve:** It is a widely accepted fact that reliable steganalysis should result in low false-alarm rates [79]. Therefore, more weightage is assigned to the AUC below the true-positive rate threshold than to the region above the threshold and then normalized in 0 and 1. The modified AUC is known as WAUC in steganalysis literature [80]. An example of WAUC is given in Figure 2.7. The weights for the area below ("red") and above ("gray") the threshold are 2 and 1, respectively. Recently, WAUC has been used as an evaluation metric for steganalysis literature [80, 81].

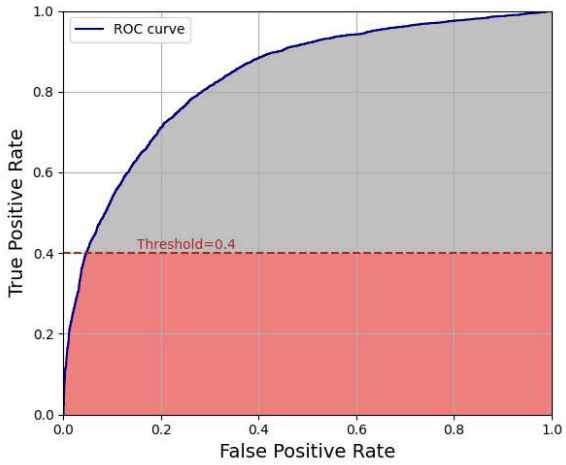


Figure 2.7: *An Illustration of WAUC.*

2.2.2 Image Quality

Any processing applied to an image may result in a loss of quality. Evaluation of image quality can be done in two ways - Subjective and Objective [82]. Subjective approaches are based on human judgment and function without specific reference parameters. Objective approaches are based on comparisons using explicit numerical criteria, and it is possible to make a variety of references, such as ground truth or prior knowledge, etc. A few examples of objective approaches used in this work to evaluate the quality of stego images are average *Peak Signal to Noise Ratio* (PSNR), the average *Structural Similarity Index* (SSIM), the average *Multi-scale Structural Similarity Index* (MS-SSIM) [1], and the average *Universal-Image-Quality Index* (UQI) [83], and the average *Visual Information Fidelity* (VIF) [84]. These metrics are calculated between the original cover images and the corresponding generated stego images.

- **PSNR:** Consider 8-bit grayscale image x and distorted version of that image y with size $M \times N$. The *Peak Signal to Noise Ratio* (PSNR) between x and y is calculated as:

$$PSNR(x,y) = 10log_{10}\left(\frac{peak^2}{MSE(x,y)}\right), \tag{2.4}$$

where *peak* denotes the maximum possible intensity (for 8-bit grayscale, $peak = 255$) and $MSE(x,y) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(x_{ij} - y_{ij})^2$

- **SSIM:** The *Structural Similarity Index* (SSIM) is another metric for image quality assessment, which is based on the degradation in structural information. The SSIM models image distortion as combination of three components - distortion in luminance (l), contrast distortion (c), and loss of correlation (s). The SSIM between image x and y is defined as:

$$SSIM(x,y) = l(x,y).c(x,y).s(x,y), \tag{2.5}$$

where

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

$l(x,y)$ measures the closeness of mean luminance of two images. $l(x,y)$ is maximum ($= 1$) when $\mu_x = \mu_y$. $c(x,y)$ compares the contrast of two images. It is maximum ($= 1$) when $\sigma_x = \sigma_y$. $s(x,y)$ compares the structure using correlation coefficient between two images x and y. The SSIM$\in [0,1]$, where 0 implies no correlation between images, and 1 implies $x = y$. C_1, C_2, and C_3 are positive constants used to avoid zero denominator.

- **MS-SSIM:** The *Multi-scale Structural Similarity Index* (MS-SSIM) takes the original image and the distorted image as input and iteratively applies low-pass filtering and downsamples the image by a factor of 2. The original image is indexed at scale 1 and the maximum scale M. At i^th scale, the contrast comparison and structure comparison denoted as $c_i(x,y)$ and $s_i(x,y)$, respectively. The luminance is compared only at scale M and is denoted by $l_M(x,y)$. The MS-SSIM is computed by combining these terms

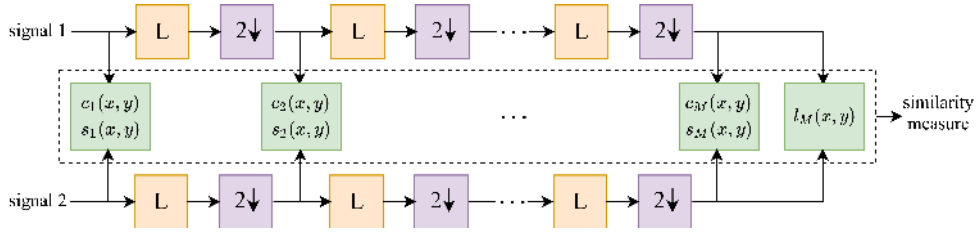


Figure 2.8: *MS-SSIM evaluation system. L: low pass filtering and 2 ↓: downsample by factor of 2 [1].*

as:

$$MS\text{-}SSIM(x,y) = [l_M(x,y)]^{\alpha_M} . \prod_{i=1}^{M} [c_i(x,y)]^{\beta_i} [s_i(x,y)]^{\gamma_j} \tag{2.6}$$

The relative importance of different components is tuned by α_M, β_i, and γ_j.

- **UQI:** The *Universal-Image-Quality Index* (UQI) is defined as

$$Q = \frac{4\sigma_{xy}\bar{x}\bar{y}}{(\sigma_x^2 + \sigma_y^2)[(\bar{x})^2 + (\bar{y})^2]}, \tag{2.7}$$

where $x = x_j | j = 1, 2, ..., N$ $y = y_j | j = 1, 2, ..., N$ are original and test images, respectively. $\bar{x} = \frac{1}{N}\sum_{j=1}^{N} x_j$, $\bar{y} = \frac{1}{N}\sum_{j=1}^{N} y_j$, $\sigma_x^2 = \frac{1}{N-1}\sum_{j=1}^{N}(x_j - \bar{x})^2$, $\sigma_y^2 = \frac{1}{N-1}\sum_{j=1}^{N}(y_j - \bar{y})^2$, and $\sigma_{xy} = \frac{1}{N-1}\sum_{j=1}^{N}(x_j - \bar{x})(y_j - \bar{y})$. The range of Q belongs to $[1, 1]$. The best value of $Q = 1$ is achieved when $y_j = x_j$, $\forall_i$. The UQI can also be written as the product of three different factors: decay in correlation, distortion in luminance, and distortion in contrast as follows:

$$Q = \frac{\sigma_{xy}}{\sigma_x \sigma_y} . \frac{2\bar{x}\bar{y}}{(\bar{x})^2 + (\bar{y})^2} . \frac{2\sigma_x \sigma_y}{\sigma_x^2 + \sigma_y^2} \tag{2.8}$$

- **VIF:** The *Visual Information Fidelity* (VIF) is a metric that measures the information fidelity for the whole image in a statistical model of the HVS [84]. The VIF uses two variables. The first is the statistics between the initial and the final stage of the visual channel without distortion. The second is the mutual information between the input of the distortion block

and the output of the visual block. The VIF is estimated for a collection of $N \times M$ wavelet coefficients from each sub-band as follows.

$$VIF = \frac{\sum_{i \in subbands} I(C^{N,i}; F^{N,i}|s^{N,i})}{\sum_{i \in subbands} I(C^{N,i}; E^{N,i}|s^{N,i})}, \tag{2.9}$$

where $I(C^{N,i}; F^{N,i}|s^{N,i})$ and $I(C^{N,i}; E^{N,i}|s^{N,i})$ are the information that can ideally be computed by the brain from a specific wavelet sub-band in the reference and the test images, respectively.

2.3 Datasets

A variety of image datasets with different resolutions and textures are used for experimentation. Details of datasets are as follows:

- **BOSSBase:** The BOSSbase dataset [85] is initially used for assessing the performance of the steganalysis schemes competing in the BOSS (*Break Our Steganography System* challenge organized in 2010. The BOSSbase dataset comprised 10,000 raw grayscale images taken from seven different digital cameras. Each image in Bossbase has a dimension of 512×512. These images are generally used as the cover image to train and test a steganalysis system.

- **BOWS2:** The BOWS2 dataset [86] is a popular dataset used in the *Break Our Watermarking System* (BOWS) contest to assess the performance of watermarking systems. BOWS2 is later also used to train and test steganalytic detectors. The BOWS2 dataset contains 10,000 grayscale images, each with a dimension of 512×512.

- **Imagenet:** The Imagenet [7] is a large-scale dataset designed for the ILSVRC started in 2010 for object recognition research. The dataset contains ≈ 1 *million* images of different sizes and 1,000 object classes. Apart

from object recognition, the Imagenet dataset is also used in various computer vision applications for the training and testing of models.

- **Microsoft COCO:** The *Common Objects in COntext* (COCO) [8] is a large-scale dataset released by Microsoft for object detection, segmentation, and image captioning tasks. It contains 330K images, with 80 object categories and five captions per image. Recently, the Microsoft COCO dataset has also been used in other computer vision applications.

- **DIV2K:** The *DIVerse 2K* (DIV2K) [87] dataset consists of 1,000 high-definition images with large diversity contents used in the NITRE challenge CVPR 2017 and 2018 for super-resolution, image dehazing tasks.

2.4 Summary

In this chapter, background concepts on the *Convolutional Neural Network* (CNN) have been presented. These concepts are used to design different kinds of steganalysis schemes proposed in the later chapters. In addition, evaluation metrics used to evaluate the methods and the datasets are presented in this chapter.

With this background, this thesis's first contribution will be discussed in the next chapter, where the first method mounts attack the *Clustering Modification Directions* (CMD) steganography and the second method performs the steganalytic detection by learning denoising kernel to replace filter used in preprocessing step.

Chapter 3

Breaking CMD Steganography and Kernel Learing for Steganalysis

The content-adaptive steganography methods such as HUGO [6], WOW [5], S-UNIWARD [15], etc., use different kinds of filters to identify the predictable and unpredictable regions and a distortion function that assigns high embedding to the unpredictable and low to the predictable areas. However, it has been observed that these algorithms distribute some of the steganographic embeddings to the predictable regions too. Considering this observation, Li *et al.* [36] proposed a steganography method called *Clustering Modification Directions* (CMD) algorithm. CMD algorithm works as a wrapper over the existing steganography algorithms. CMD method clusters the embeddings to a specific direction to improve security. When CMD wrapping is applied over the existing steganographic algorithms, a state-of-the-art steganalyzer suffers the loss of detection accuracy by $\sim 5\%$. On the other side, it has been observed that the existing steganalysis schemes use a set of fixed handcrafted high-pass filters for enhancing the stego noise while suppressing the image contents. Due to the heterogeneity of embedding algorithms, a fixed filter may not be beneficial in extracting meaningful features.

Motivated by the above observations, this chapter presents two steganalysis approaches. The first method, discussed in Section 3.1, is proposed to detect the

CMD steganography, thereby nullifying the effect of CMD. The second method, described in Section 3.2, is a deep learning-based approach that learns a filter that can replace the fixed high-pass filter (KV filter) in the preprocessing step of the steganalyzers. Learning the filter enables the steganalyzer to improve the performance of the classifier used for the detection.

3.1 Breaking CMD steganography

In this section, a novel steganalysis method called *Selective-Signal-Removal* (SSR) is proposed to attack the CMD steganography. The SSR method's preprocessing step is motivated by the selective nature of the CMD in terms of embedding locations. In the SSR scheme, the correspondence between the texture distribution and the CMD scheme's embedding locations are exploited to attack CMD steganography. This scheme allows the feature extraction from the image regions directly affected by the data embedding rather than from the whole image.

3.1.1 CMD steganography

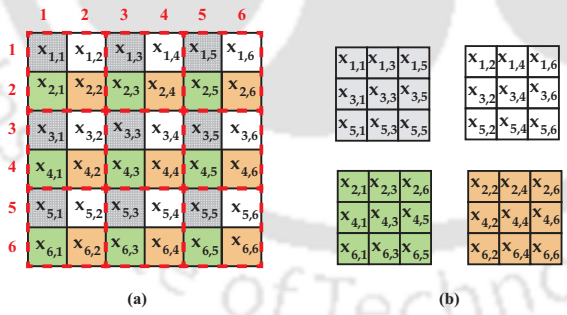


Figure 3.1: *An example of decomposition of an image into 4 non-overlapping sub-images.* ***(a)*** *The original image pixel-grid in which red dotted boundary indicates a sub-block of the images, the four elements in each sub-block assigned to different sub-images.* ***(b)*** *4 sub-images.*

The CMD algorithm starts with decomposing the cover image into non-overlapping sub-images with size $L_1 \times L_2$, for $L_1, L_2 \geq 1$. The given data payload with m bits is also divided accordingly into segments. Each segment size is $m/L_1.L_2$ (for

fixed block-size). An embedding order is chosen to embed in the sub-images (e.g., horizontal zig-zag order). Initially, the first sub-image is initialized with the cover image, and the difference image, D, is calculated between the stego and cover image ($stego - cover$). Next, the initial cost of the stego image is assigned using any of the existing steganography schemes, such as S-UNIWARD, WOW, or HILL. The estimated difference image is used to determine the direction (here, direction denotes the positive or negative change in pixel intensity) of pixel changes. For the remaining blocks, the distortion function is designed in such a way that pixel modifications follow similar directions. If a pixel has more neighbors modified to $+1$, then its cost for positive embedding is reduced by a constant factor of α. Similarly, if a pixel has a majority -1 modified neighbors, its cost for negative embedding is reduced by α. This change in the distortion function ensures that the embeddings in the same directions are clustered together. Figure 3.1 shows the decomposition of a 6×6 image. The four colors denote the four subsampled sub-images from the original image.

The CMD algorithm embeds the data, preserving the inter-pixel dependencies by changing all pixels in a local cluster in the same direction. If we follow the three steps of steganalysis, namely: high-pass filtering, feature extraction, and classification, CMD effectively prevents good quality features from being extracted. A local cluster of modified pixels has the same response to a high-pass filter in both cover and stego images; this results in inter-pixel dependencies more diminutive than that in non-CMD embedding. All the handcrafted features like state transition matrices of SPAM [13] and co-occurrence matrices of SRM [14] rely on the disruptions in pixel dependencies to generate rich feature vectors. Therefore, SPAM and SRM cannot steganalyze the CMD scheme with usual accuracy.

A detailed study of the CMD embedding patterns reveal the following critical observations:

1. The CMD and non-CMD algorithm (pure WOW, HILL, S-UNIWARD) strive to maximize the embedding capacity in pixels that are surrounded

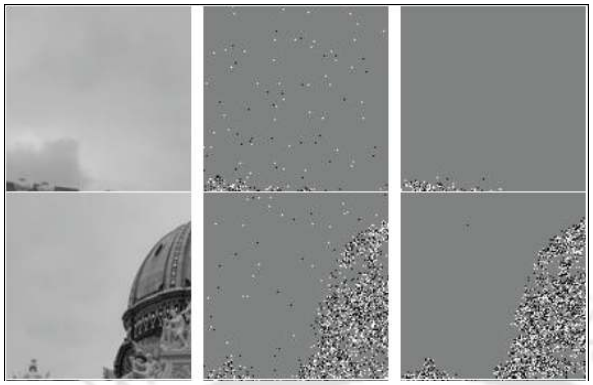


Figure 3.2: *Examples of embedding by non-CMD and CMD approach. The first column shows cover images; the second column shows the simple S-UNIWARD embedding (with 0.5 bpp) locations in the corresponding image and the third column shows the embedding locations in the corresponding image in the case of CMD-S-UNIWARD (with 0.5 bpp) steganography.*

by texture-rich areas of the image.

2. It is observed that CMD embedding follows a typical distribution as it tries to embed in neighboring areas with similar embedding directions. As a result, it may be possible that data is embedded in low-frequency zones, but it is not statistically visible. This typical nature of embedding helps CMD to be statistically undetectable against conventional steganalysis, which assumes steganographic embedding usually chooses a high-frequency zone for better covertness. However, it is experimentally observed that most of the visible CMD embedding is clustered in high-texture regions. (Please refer to Figure 3.3).

3. The CMD algorithm forms clusters of pixels that are modified in the same direction. Such groups form super-pixels in the difference image corresponding to the image.

The aforementioned observations can be seen in Figure 3.2.

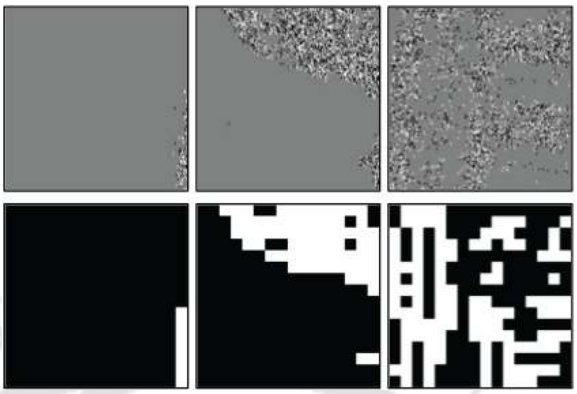


Figure 3.3: *Observations of CMD embedding effect through the proposed heuristic function and assignment algorithm. First row: Examples of the difference (stego − cover) for CMD-S-UNIWARD embedding with 0.5 bpp where white and black pixels represent +1 and −1 embedding respectively. Second row: White blocks denote the region of the image above the threshold T, and black indicates regions below the threshold T.*

3.1.2 Selective Signal Removal (SSR)

Motivated by the above observations, the relation between image texture and embedding locations is investigated. The *Discrete Cosine Transform* (DCT) [88] has been widely used in the area of image processing to capture the texture variation in an image [89,90]. In the proposed scheme, DCT is used to find the high texture regions in an image. The sum of the squared coefficients (excluding the DC component) is used to capture the texture variation in a block effectively. A threshold for each image is investigated through a comprehensive set of experimentation to locate the possible regions (relevant regions) of CMD embedding. It has been observed that CMD tends to embed in high texture regions, which is exploited in this work. This observation is also shown in Figure 3.3 and experimentally validated in the subsequent sections.

The *Selective-Signal-Removal* (SSR) technique, as the name suggests, essentially seeks to remove pixels from irrelevant regions (regions having less probability for CMD embedding) and allows only relevant regions for generating discriminative features. The idea is to eliminate such irrelevant regions of the image

to estimate a more useful feature. The SSR scheme divides the image into non-overlapping blocks (super-pixel) of size $B \times B$. A heuristic function determines whether a given block is relevant or not and assigns 0 (non-relevant) and 1 (relevant) accordingly. This assignment process generates a bit-map image. The bit-map is then upscaled to match the dimensions of the given image. Finally, the Hadamard product is used to map the bit-map on the original image to form a new image I'. The set of new images forms a new dataset. Further, the SRM is trained on this new dataset. The efficiency of the heuristic function and assignment algorithm depends on its accuracy to find the embedded regions within an image. The block diagram of the SSR scheme is depicted in Figure 3.4.

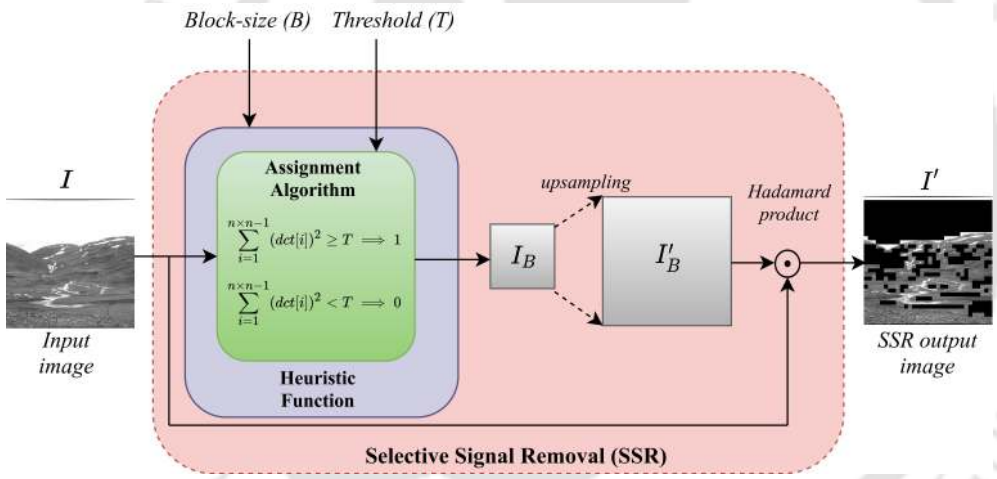


Figure 3.4: *Block diagram of proposed SSR scheme*

3.1.2.1 Heuristic Function

The heuristic function uses an assignment algorithm to determine whether a super-pixel is relevant or irrelevant based on the threshold value (T). The heuristic function h is as follows:

$$h : (I,\ B,\ T) \longrightarrow I_B \tag{3.1}$$

The dimensions of mask I_B depend on block size B. For example, given an image I of dimensions 512×512 and block size of 4×4, we get I_B of dimensions

128×128 (as $512/4 = 128$) as each super-pixel represents a 4×4 square of pixels. After the generation of I_B, it is up-sampled to match the dimension of the original image, say I'_B.

3.1.2.2 Assignment Algorithm

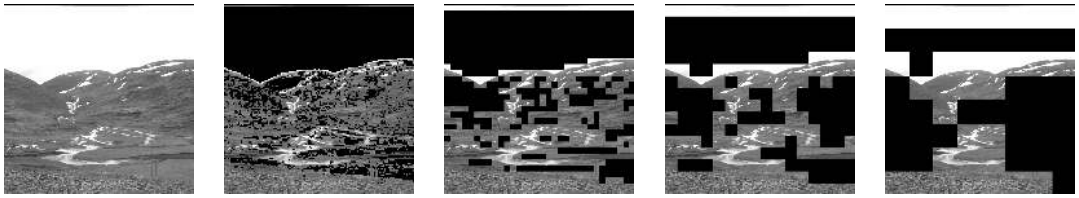
The assignment algorithm discriminates between the relevant and irrelevant super-pixels by means of assigning values 0 and 1. The assignment algorithm assigns 1 (*relevant*) and 0 (*irrelevant*) to a super-pixel using a threshold value T. The super-pixels above this threshold are assigned 1, and the others are assigned 0. The assignment algorithm uses the following equation:

$$\begin{aligned} \sum_{i=1}^{(n\times n)-1} (dct[i])^2 \geq T \Rightarrow assign\ 1 \\ \sum_{i=1}^{(n\times n)-1} (dct[i])^2 < T \Rightarrow assign\ 0, \end{aligned} \tag{3.2}$$

where $dct[i]$ denotes the DCT of super-pixel i in the bit-map, T represents the threshold, and $n \times n$ denotes the size of the bit-map.

3.1.2.3 Threshold

The threshold parameter, T, divides an image into embedding and non-embedding zones. This parameter is image-dependent, i.e., it may vary from image to image. In order to find out the threshold value for an image, the following procedure is followed: (i) Initially, the image is divided into the super-pixels of size $B \times B$. (ii) Each super-pixel is subjected to the DCT, and the squared sum of all AC co-efficients is computed, which results in a new image with size $n/B \times n/B$ (since the sum of squared DCT value gives a single value). (iii) It is experimentally found that the *median* works best as a threshold for segmenting the image into embedding and non-embedding zones.



(a) cover image (b) $B = 4$ (c) $B = 16$ (d) $B = 32$ (e) $B = 64$

Figure 3.5: *Illustration of the effect of block-size (B) selection on the heuristic function output.*

3.1.2.4 Identification of Optimal Block Size

Block size B is an essential parameter for the heuristic function h, as it defines the number of pixels in each super-pixel. For example, $B = 4$ means that the heuristic function considers the super-pixel of size 4×4. While up-sampling from I_B to I'_B, the whole block is allocated the corresponding boolean form, as is I_B. Thus in the final image, I' is obtained by applying Hadamard product between the original image I and I'_B, i.e., $I' = I \odot I'_B$. The block size manifests as the *coarseness* of the removed image signals. Figure 3.5 shows how block size affects the appearance of the final image I'.

In order to find the best suitable block size, we have defined the *Block Correlation* (B_{cor}) as the ratio of the total number of non-zero pixels in the difference image of cover and stego processed by the SSR scheme and without the SSR scheme. Let X and Y denote a cover and CMD-stego images, respectively, and let X' and Y' indicate a cover and CMD-stego image after processed by the SSR scheme. Let us denote the difference between the cover and the CMD-stego image processed by the SSR scheme as $D_{SSR} = Y' - X'$ and the difference between the cover and CMD-stego as $D = Y - X$. Then B_{cor} is given by eq. (3.3).

$$B_{cor} = \frac{no.\ of\ non\text{-}zero\ pixels\ in\ D_{SSR}}{no.\ of\ non\text{-}zero\ pixels\ in\ D} \tag{3.3}$$

A high value of B_{cor} implies that the image I' processed by SSR has many pixels that are different in cover and stego, which is obvious since many irrelevant super-pixels are removed (which are not affected by the embedding). In order to

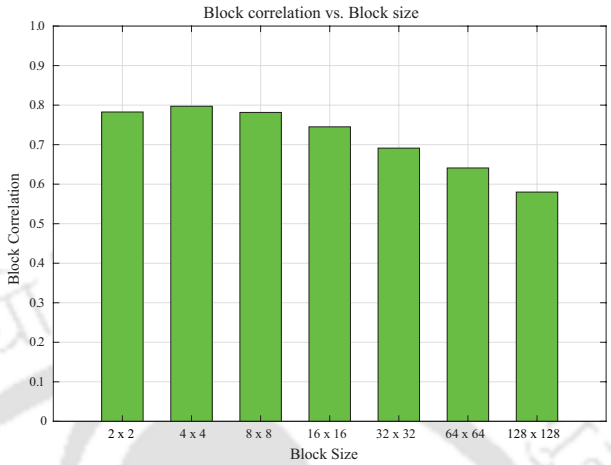


Figure 3.6: *Illustration of the effect on Block Correlation (B_{cor}) with different block size.*

select the most affected region of the image by the embedding, a high block correlation is preferable. In order to justify this claim, B_{cor} is compared for different block sizes over the entire dataset (10,000 images of *BossBase* [85]) with CMD-S-UNIWARD [36]. Figure 3.6 shows the variation of B_{cor} over different block sizes. The bar graph in Figure 3.6 shows an initial rise from $B = 2$ to $B = 4$ and then falls monotonously up to $B = 128$. This observation is intuitive due to the obvious geometrical constraints on large block sizes and its inability to represent finer distributions of embedding locations. For example, in the experiments, a block size of 128 makes it inappropriate to choose 8 out of 16 boxes (image dimension $= 512 \times 512$) such that the correlation is high. This is because there are only sixteen 128×128 blocks in the image, and picking 8 out of the 16 such that they map the embedding pixels perfectly is unlikely. However, if the block size is very small, then the DCT coefficients may fail to capture the texture variation. In that case, the edges and corners in the image have a high sum of squares DCT values even though they may not belong to a texture-rich region. This can be observed for $B = 2$, where the B_{cor} is 0.782606 as compared to an average correlation of 0.797211 for $B = 4$. SRM captures the four-dimensional inter-pixel dependencies of the image. Therefore, the SRM features consider every four consecutive pixels

Table 3.1: *Comparison of the classification error (in %) of SRM and proposed SSR-SRM scheme for different block sizes on the different payloads. The SSR-SRM has lower classification error for the block-size* 16×16. *The lower classification errors are shown in boldface.*

Payload (bpp)	SRM	SSR-SRM 4×4	SSR-SRM 8×8	SSR-SRM 16×16	SSR-SRM 32×32	SSR-SRM 64×64
0.5	20.79	19.42	16.99	**16.51**	16.69	17.14
0.4	25.46	24.22	21.02	**20.52**	20.88	21.89
0.3	30.65	29.18	25.77	**25.53**	25.74	27.04
0.2	35.34	34.86	32.66	**31.24**	32.47	34.12

and their relation to each other. Thus, a smaller block size may fail to model the inter-pixel dependency used in SRM, whereas the correlation with embedded pixels (non zero entries of D) is lower for large block sizes. Intuitively, a relatively smaller block can model ± 1 pixels of D more accurately. A larger block preserves the inter-pixel dependencies (that are modeled as the co-occurrence matrix in the SRM), which means that the amount of spatial distortion introduced in the image is relatively less for a large block size as compared to a smaller block. This trade-off implies that intermediate block size should be chosen for the assignment algorithm. In order to find an optimum block size, the proposed SSR technique is tested on the CMD-S-UNIWARD at embedding rates of 0.2, 0.3, 0.4, and 0.5 bpp. The classification test error of the scheme is shown in Table 3.1. Block size of 16×16 dominates all other block sizes across all the embedding rates and may be the optimal parameter for our proposed scheme. The relation between the block correlation and the classification accuracies for different block-sizes are shown in Figure 3.7. From Figure 3.7, it can be observed that the highest test accuracy is achieved for the block size 16×16, which has a block correlation of ~ 0.74. This result is consistent with our hypothesis that an intermediate block size provides a balance between block correlation with embedding locations and the effective population of SRM features.

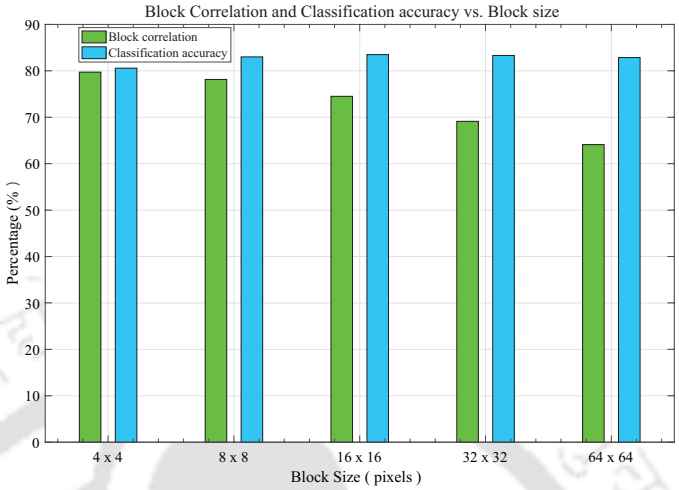


Figure 3.7: *Classification accuracy (Green) and block correlation (Blue) with different block size.*

3.1.2.5 Discussion on Effectiveness of the SSR

The SSR technique generates features subjected to *Fischer Linear Discriminant* (FLD) classifier [91]. FLD is a useful classifier if the distance between class means is high and the intra-class variance is low. This is shown by making a valid assumption that entries of the SRM features for cover and stego images can be modeled as random variables independent of each other. The SRM feature is a high-dimensional array where each element is a normalized co-occurrence of a particular pattern in the high-pass residual. Let C be a random variable vector that denotes an SRM feature. The SRM feature is a co-occurrence matrix that stores how many times a particular pixel sequence occurs in the residual. For example, $SRM[w][x][y][z]$ is a number that represents the number of times the pixel values of w, x, y, and z occur consecutively (either vertically or horizontally). Now each of these patterns can appear either in the area of the image marked as 0 or in the area marked as 1, and this assignment is independent of each other. And the sum of these two will give the value of $SRM[w][x][y][z]$, which is given by C. Therefore, C can be written as $C_0 + C_1$, where C_0 denotes the co-occurrence due to 0 labeled part and C_1 indicates the part of image I labeled as 1 (labeling

is done by the assignment algorithm); similarly, X_0 and Y_0 denote the cover and stego labeled as 0 and X_1 and Y_1 denote the cover and the stego labeled as 1. For a cover-stego pair (say X and Y), C is as follows:

$$\begin{aligned} C_X &= C_{X_0} + C_{X_1} \\ C_Y &= C_{Y_0} + C_{Y_1} \end{aligned} \tag{3.4}$$

Since C_0 and C_1 are independent, in the ideal case, the variances of C_X and C_Y can be written as:

$$\begin{aligned} Var_{ideal}(C_X) &= Var(C_{X_0}) + Var(C_{X_1}) \\ Var_{ideal}(C_Y) &= Var(C_{Y_0}) + Var(C_{Y_1}) \end{aligned} \tag{3.5}$$

With the SSR technique, the signal of all the parts of the image that are labeled as 0 (by defining $I' = I \odot I'_B$) are removed. Now C has to be written as C_1 (instead of $C_0 + C_1$ as the 0 labeled part has been removed and contributes 0 to the value). Therefore, X and Y can be written as:

$$\begin{aligned} C_X &= C_{X_1} \\ C_Y &= C_{Y_1} \end{aligned} \tag{3.6}$$

Similarly, the variances of C_X and C_Y can be written as:

$$\begin{aligned} Var_{ssr}(C_X) &= Var(C_{X_1}) \\ Var_{ssr}(C_Y) &= Var(C_{Y_1}) \end{aligned} \tag{3.7}$$

Therefore, from eq. (3.5) and (3.7) we get:

$$\begin{aligned} Var_{ssr}(C_X) &\leq Var_{ideal}(C_X) \\ Var_{ssr}(C_Y) &\leq Var_{ideal}(C_Y) \end{aligned}$$

This explains the reduction in variance for cover and stego. It is also empirically verified that the variance decreases to 26,394 from 34,671 dimensions for the cover images class (76.12%). For the stego class, variance decreases to 34,061 from 34,671 (SRM feature dimension) dimensions (98.24%). The observations are in agreement with our hypothesis. These observations are taken for SRM and SSR-SRM features extracted from CMD-S-UNIWARD at 0.4 bpp embedding rate.

3.1.3 Experimental Study

3.1.3.1 Experimental setup

Experiments are performed using the *BossBase v1.0* dataset [85] with 10,000 grayscale images with size 512×512. We performed classification on the dataset using the ensemble classifier (with FLD classifiers as base learners) on the SRM features. Randomly selected 5,000 images are used for training and the rest for testing with five-fold cross-validation. Experiments are performed using the SSR preprocessing on state-of-the-art steganography algorithms: CMD-S-UNIWARD, CMD-WOW, and CMD-HILL. Parameters for CMD are kept as $L = 2$ and $\alpha = 9$ with the embedding done in a row by row order for different subsamples. In order to get a detailed picture of the improvements due to SSR, embedding rates used are 0.2 bpp, 0.3 bpp, 0.4 bpp, and 0.5 bpp. For each of the above, the optimal block size, $B = 16$, is used in the heuristic function h given in eq. (3.1). The steganalytic classification error (P_E) is calculated using eq. (2.3). The probabilities are represented on a scale of 100 (as percentage error) in the results.

3.1.3.2 Comparison with the State-of-the-art Steganalyzers

This subsection compares the performance of the proposed scheme against the CMD-S-UNIWARD, CMD-WOW, and CMD-HILL in terms of error detection accuracy while classifying between stego and cover image. The detection error is reported in terms of percentage (%) error [92]. Table 3.2 compares the proposed scheme's performance with SRM [14] against CMD-S-UNIWARD, CMD-WOW, and CMD-HILL steganography, respectively. The proposed scheme clearly shows a significant improvement over the state-of-the-art. Finally, Table 3.3 shows how the proposed SSR algorithm mostly nullifies the effect of CMD wrapping on a steganography algorithm. We can achieve almost the same accuracy as an SRM based classifier attain on simple (without CMD embedding) S-UNIWARD steganography.

Table 3.2: *Comparison of the classification error (in %) of proposed SSR-SRM scheme with SRM on CMD-WOW, CMD-SUNIWARD, and CMD-HILL.*

Embedding	Payload (bpp)	SRM	SSR-SRM
CMD-WOW	0.5	20.85	**16.10**
	0.4	25.48	**19.84**
	0.3	29.82	**24.59**
	0.2	34.66	**31.15**
CMD-SUNIWARD	0.5	20.79	**16.51**
	0.4	25.46	**20.52**
	0.3	30.65	**25.53**
	0.2	35.34	**31.24**
CMD-HILL	0.5	25.75	**20.42**
	0.4	30.32	**24.06**
	0.3	35.96	**29.20**
	0.2	39.60	**35.14**

Table 3.3: *Comparison of classification error (in %) between SRM applied on simple S-UNIWARD (SRM on S-UNIWARD) and SRM applied on SSR processed CMD-S-UNIWARD (SSR-SRM).*

Embedding Rate	SRM on S-UNIWARD	SSR-SRM on CMD-S-UNIWARD
0.5	15.46	**16.51**
0.4	20.69	**20.52**
0.3	25.65	**25.53**
0.2	31.34	**31.24**

3.2 Kernel Learning for Steganalysis

Recently, deep learning-based methods such as CNNs have been successful for steganalysis. These methods, in general, follow a three-step process- *preprocessing*, *feature learning*, and *classification*. However, the last two steps, i.e., feature learning and classification, are usually performed with a single deep learning network. Unlike a simple binary classification where both classes' features vary, steganalysis is challenging because the stego and cover images are visually identical. Therefore, classifiers are trained on noise content despite image content. In order to train on noise content, state-of-the-art steganalyzers use a set of high-pass filters to suppress the image content and enhance noise content. The feature learning is carried out by the convolutional layers. The learned features are used by fully connected layers for classification.

It has been observed in the literature presented in Section 1.4.2 that most of the existing CNN-based methods used a fixed high-pass filter for preprocessing of images with the assumption that steganographic noise is a typical high-frequency noise. In addition, most of the embedding schemes generally tend to embed in high-frequency image zones as high-frequency content helps to mask the steganographic noise. Moreover, some of these approaches used a deeper network for classification, which may lead to overfitting training data. Considering the heterogeneity among embedding parameters, like image statistics, embedding rate, embedding strategies, etc., fixed kernel-based noise extraction may not always be optimal. Besides, a deeper network for classification may lead to overfitting. With the above line of thought, the main contributions of the proposed work are two-fold:

1. A filter kernel is learned using a CNN to extract the stego noise more accurately in preprocessing step.

2. An existing [2] CNN model is employed with some minor modifications as the steganalytic classifier.

3.2.1 Proposed Work

The proposed work is comprised of two modules. In the first module, a denoising CNN is trained, where the stego images are given as input and the corresponding cover images as ground-truth. The learned denoising CNN is used to extract the steganographic noise residual, which is used to train the steganalytic classifier to detect the stego images.

3.2.1.1 Denoising Kernel Learning using CNN

The proposed denoising CNN (DCNN) consists of a single convolution layer with 16 kernels with a size 5×5[1]. The proposed network has no pooling layer because, in general, the stego noise is a relatively weak signal, and applying a pooling operation may discard the stego noise. The stride is fixed to 1 so that it convolves over the entire image without any padding. The proposed DCNN predicts a denoised cover image from a stego image. The learned kernels of DCNN are used with a classification network for steganalytic detection. The architecture of the proposed DCNN is shown in Figure 3.8.

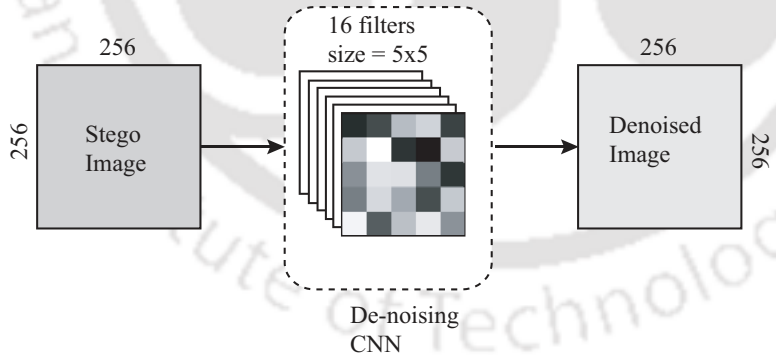


Figure 3.8: *Architecture of CNN used for denoising CNN(DCNN).*

The DCNN takes an image (cover or stego) of size 256 × 256 and predicts the corresponding cover image. The predicted cover image is subtracted from

[1]In order to find the suitable kernel size for denoising, the extensive experiment has been carried out with different filter sizes (e.g., 3 × 3, 5 × 5 and 7 × 7). The 5×5 filters are found to be suitable as denoising filters.

the input image to obtain the steganographic noise. The DCNN takes an input image I and predicts denoised image $\hat{I}$ as output.

$$\hat{I} = f_{DCNN}(I), \tag{3.8}$$

where the function f_{DCNN} is the filter learned by DCNN. A pixel-wise difference between the input image (stego) and the denoised image (predicted cover) is calculated as the steganographic noise residual $\hat{R}_{i,j}$.

$$\hat{R}_{i,j} = I_{i,j} - \hat{I}_{i,j}, \tag{3.9}$$

where $\hat{R}_{i,j}$, $I_{i,j}$, $\hat{I}_{i,j}$ are noise residual, input image, and denoised image (predicted cover by DCNN), respectively, at pixel (i, j).

3.2.1.2 Steganalytic Classifier

In the second module of this work, a shallower CNN architecture is employed as a steganalytic classifier. It consists of two convolutional layers and two fully connected layers, and a two-way softmax layer. In order to come up with this architecture of the steganalytic classifier, an extensive set of experiments is conducted with a varying filter size of each layer from the set $\{3 \times 3, 5 \times 5, 7 \times 7\}$. The number of hidden units in fully connected layers is varied from the set {4096, 2048, 1000, 750}. The fully connected layers with 1000 neurons have similar steganalytic classification errors compared to that of 750 neurons. However, the use of 750 neurons has reduced the number of training parameters. The description of the final steganalytic classification module is given in Figure 3.9.

The first layer of the classification CNN takes noise residual with 256×256 dimensions as input. The first convolutional layer is composed of 64 filters, each of size 7×7. The first layer's output is 64 feature maps of size 128×128, which are used by the second layer as input. The second layer consists of 16 filters of size 5×5 to extract more detailed features. The output of this layer is 16 feature maps of size 64×64, which is further fed to the fully connected layers. The fully connected layers are consist of 750 neurons each. Finally, a softmax layer is used

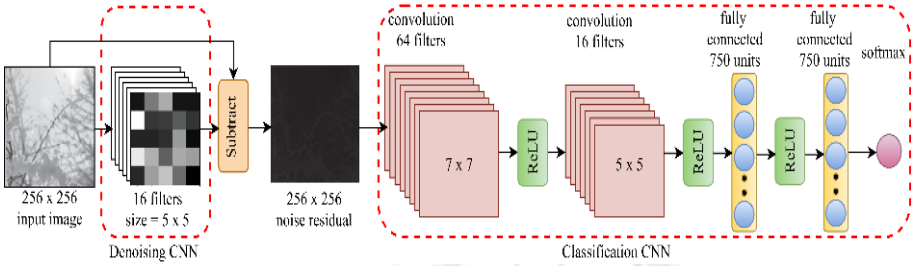


Figure 3.9: *The architecture of the proposed steganalyzer: The left part represents Denoising CNN (DCNN), and the right part shows the classifier network.*

for binary classification (cover and stego). The ReLU activation is applied to each layer of the classification network. Moreover, dropouts [93] (with a probability of 0.8) are applied to the second convolution layer and the hidden layers to mitigate network overfitting. The model does not include any pooling layers as it may suppress the weak stego noise.

3.2.1.3 Training

The training of DCNN and the classifier is carried out on BOSSBase v1.0 [85] and BOWS2 Ep.3 [86] datasets. The BOSSBase and BOWS2 datasets are having 10,000 gray-scale images. Each image is split into four sub-images from the center, each with a size 256×256 to increase the dataset size. The creation of sub-images is shown in Figure 3.10. Therefore, each of the BOSSBase and BOWS2 datasets is having 40,000 images. We call these new datasets as *cropped-BOSSBase* and *cropped-BOWS2* datasets, respectively. Both networks are trained and tested in three scenarios.

Scenario-1: Under this scenario, 15,000 pairs (15,000 cover and 15,000 stego) of images from cropped BOSSBase dataset are used to train DCNN. The validation of DCNN is done on 20% of training pairs. The Steganalytic classifier is trained and validated (with 20% of training data) using another 15,000 pairs. A set of 10,000 pairs are kept aside for testing of the complete network.

Scenario-2: The DCNN under this scenario is trained and validated (with 20% of training pairs) using 30,000 pairs (30,000 cover and 30,000 stego) images

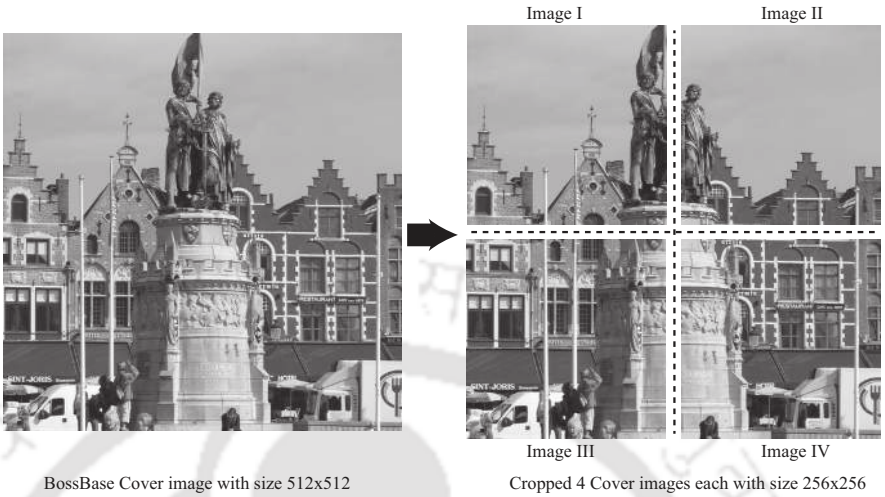


Figure 3.10: *Division of images (Original size* $= 512 \times 512$ *image) into four sub-images each of size* 256×256.

from cropped BOSSBase v1.0 [85]. The steganalytic classifier is trained with 30,000 pairs of cropped BOWS2 [86] and validated (with 20% of training pairs), and testing is done using 10,000 pairs of images from cropped BOSSBase v1.0 and 10,000 pairs of images from cropped BOWS2 dataset.

Scenario-3: Under this scenario, mixed datasets are formed with 2,500 images each using S-UNIWARD [15], WOW [5], and HUGO [6] with payloads 0.2, 0.3, 0.4 and 0.4 bpp, total $3 \times 4 \times 2,500 = 30,000$ pairs for both, cropped-BOSSBase and cropped-BOWS2 dataset separately, while leaving 10,000 pairs untouched for testing. Both the CNNs are trained and tested in two ways:

(i) DCNN is trained and validated on 30,000 (20% validation) mixed-cropped-BOSSBase and Steganalytic classifier on 30,000 (20% validation) mixed-cropped-BOWS2.

(ii) DCNN is trained and validated on 30,000 (20% validation) mixed-cropped-BOWS2 and classifier on 30,000 (20% validation) mixed-cropped-BOSSBase.

Training parameters of DCNN All the kernels of the DCNN are initialized to a random normal distribution with $\mu = 0$ and $\sigma = 0.02$. During training, the Adagrad optimization algorithm [94] is used with a learning rate of 0.001.

Each batch consists of 150 images (75 cover and 75 stego images). The mean-squared loss is used as a cost function. The network is trained until it converges ($\approx$ 80 epochs), and the model with the best validation accuracy is chosen for the evaluation of the model.

Evaluation of DCNN The kernel's weight learned by the DCNN is similar to various kernels of SRM. Figure 3.11 depicts the visualizations of (16 filters) of the trained DCNN. It is clear that there is no dead filter [3] (filter with almost white features is referred to as the *dead filter*), and all filters have learned to extract slightly different noise characteristics.

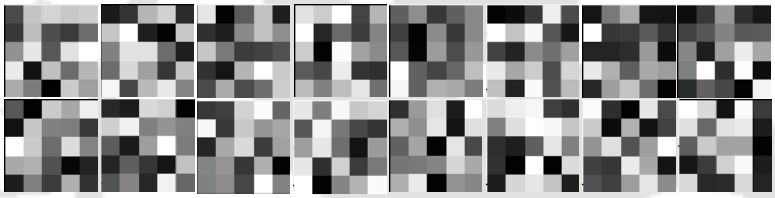


Figure 3.11: *Visualization of 16 learned kernels weights of size 5×5 of DCNN.*

To show the effectiveness of DCNN, we evaluated the noise residuals resulted from the high-pass filter and DCNN using 10,000 test images from the cropped-BOSSBase dataset using *Peak Signal to Noise Ratio* (PSNR) and *Structural Similarity Index* (SSIM).

(i) *Cover image prediction using high-pass filters*: First stego image is convolved with the high-pass filter (KV filter) eq. (1.5), which results in the noise residual. This noise residual is subtracted from the stego image, which yields a cover image predicted by the KV filter.

(ii) *Cover image prediction using DCNN*: DCNN is trained to predict the cover image from a given input image.

The PSNR and the SSIM between the original cover and cover predicted using a high-pass filter (HPF) is calculated; likewise, the PSNR and the SSIM between the original cover and cover predicted using DCNN is calculated. The

Table 3.4: *Quantitative evaluation of DCNN in terms of mean (μ) and standard deviation (σ) of PSNR (dB) and SSIM. Tabulated scores are obtained with original cover and cover image predicted by the HPF and the DCNN on different embedding configurations. P† refers to the considered set of payloads {0.2,0.3,0.4,0.5} in bpp.*

Method	Embedding scheme	Payload (bpp)	μ(PSNR)	σ(PSNR)	μ(SSIM)	σ(SSIM)
HPF	S-UNIWARD HUGO WOW	P†	30.719	10.56	0.908	0.13
DCNN	S-UNIWARD	0.5	**44.878**	8.12	**0.996**	0.01
		0.4	**41.791**	6.88	**0.994**	0.01
		0.3	**40.536**	7.34	**0.998**	0.00
		0.2	**39.103**	6.38	**0.995**	0.01
	HUGO	0.5	**42.267**	6.45	**0.994**	0.01
		0.4	**42.023**	6.13	**0.996**	0.01
		0.3	**41.557**	6.81	**0.994**	0.01
		0.2	**40.383**	7.05	**0.997**	0.00
	WOW	0.5	**43.612**	5.56	**0.996**	0.00
		0.4	**42.516**	6.29	**0.995**	0.01
		0.3	**41.592**	8.16	**0.995**	0.01
		0.2	**40.215**	8.95	**0.991**	0.02

average PSNR (μ(PSNR)), standard-deviation of the PSNR (σ(PSNR)) and average SSIM (μ(SSIM)) and standard deviation of the SSIM (σ(SSIM)) between the original cover and predicted cover of 10,000 test images for different steganographic embeddings (S-UNIWARD, HUGO, WOW) and different payloads (0.2, 0.3, 0.4, 0.5 bpp) are tabulated in Table 3.4 for both high-pass filters and proposed DCNN as the denoising tool. It has been observed that the PSNR and SSIM for the proposed DCNN are relatively higher than that of the HPF, and thus the efficacy of the proposed DCNN has been shown over conventional HPF.

Training parameters of Steganalytic Classifier This CNN is trained with the stego noise produced by DCNN. All the network layers are initialized with random normal distribution with $\mu = 0$ and $\sigma = 0.02$. The learning algorithm used is Adagrad [94] optimizer. Each mini-batch consists of 150 images (75 cover and 75 stego images). The learning rate is fixed to 0.005. The categorical cross-entropy loss [95] function is used for error calculation while training. Intuitively, the network converges faster for the images with high embedding rates than the

images with a low embedding rate while training.

3.2.2 Experimental Results and Discussion

In this subsection, the steganalytic performance of the proposed model is compared with the state-of-the-art steganalysis schemes.

Table 3.5: *Steganalytic detection error comparison of proposed scheme with SPAM [13], SRM [14], GNCNN [2], YeNet [3], and SRNet [4] against S-UNIWARD [15], HUGO [6] and WOW [5] steganography under Scenario -1. The best result is denoted using* ***boldface*** *values.*

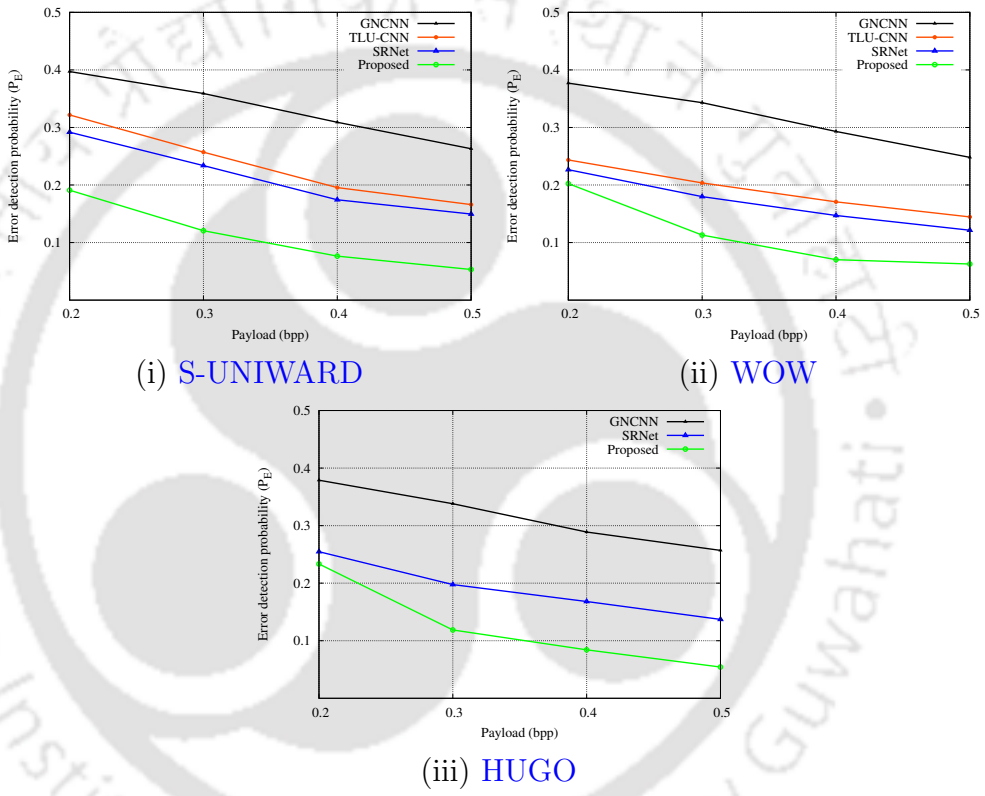
Steganography scheme	Steganalysis scheme	0.2 bpp	0.3 bpp	0.4 bpp	0.5 bpp
S-UNIWARD	SPAM	0.5189	0.4000	0.351	0.306
	SRM	0.3607	0.315	0.267	0.214
	GNCNN	0.3971	0.359	0.309	0.263
	YeNet	0.3218	0.2571	0.1955	0.1660
	SRNet	0.2917	0.2337	0.1743	0.1496
	Proposed	**0.191**	**0.1206**	**0.0764**	**0.0532**
HUGO	SPAM	0.4371	0.429	0.391	0.357
	SRM	0.3638	0.296	0.252	0.214
	GNCNN	0.3789	0.338	0.2889	0.257
	SRNet	0.2547	0.1976	0.1681	0.1371
	Proposed	**0.2334**	**0.1187**	**0.0842**	**0.0542**
WOW	SPAM	0.445	0.429	0.390	0.3661
	SRM	0.3896	0.312	0.257	0.221
	GNCNN	0.377	0.343	0.293	0.248
	YeNet	0.2435	0.2036	0.1707	0.1445
	SRNet	0.2266	0.1798	0.1471	0.1214
	Proposed	**0.2024**	**0.113**	**0.0702**	**0.0628**

The detection accuracy of the proposed scheme is compared with SPAM [13], SRM [14], GNCNN [2], YeNet [3], and SRNet [4] against three state-of-the-art spatial domain steganographic algorithms – S-UNIWARD, HUGO, and WOW. The proposed scheme is also compared with ReST-NET [16] against S-UNIWARD. The Matlab code for these steganographic schemes is taken from [96], which are single key steganography algorithms. For comparison, the SRNet model [1] is implemented in the same setting as the proposed scheme for 500000 iterations. The model with the best validation accuracy[2] has been chosen for comparison with the

[1]SRNet model code is available at http://dde.binghmton.edu/download/

[2]SRNet trained weights, test data, and codes are available at https://drive.google.com/

proposed model. The steganalytic detection performance is compared in terms of classification error (P_E) defined in eq. (2.3). The detection performance of the proposed scheme is tabulated in Table 3.5. The best result is shown using boldface. For further illustration, the results presented in Table 3.5 are graphically depicted in Figure 3.12.



(i) S-UNIWARD (ii) WOW

(iii) HUGO

Figure 3.12: *Steganalytic detection error comparison of the proposed scheme under scenario-1 with GNCNN [2], YeNet [3] and SRNet [4] against (i) S-UNIWARD, (ii) WOW [5] and (iii) HUGO [6].*

Figure 3.12 (i) and (ii) show the comparison of the proposed scheme with the state-of-the-art steganalytic schemes against S-UNIWARD and WOW respectively. Figure 3.12 (iii) represents the comparison of the proposed scheme with the state-of-the-art steganalytic schemes against HUGO. A *Receiver Operating Characteristics* (ROC) curve is also given in Figure 3.13 to support the

`open?id=1MxObzvnkFSSGR4gcfcl_Esqp27kSk5TN`

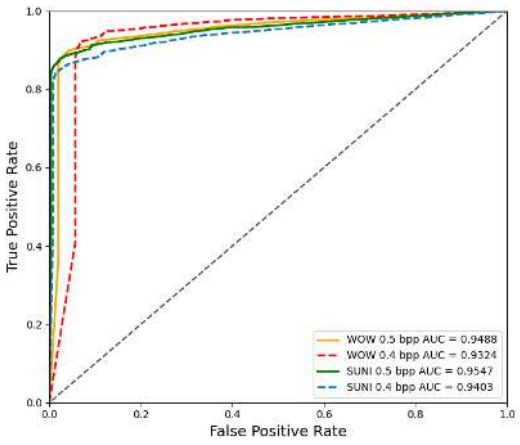


Figure 3.13: *ROC curve for the steganalytic detection of the proposed scheme for WOW and S-UNIWARD embedding with 0.4 and 0.5 bpp.*

performance of the proposed steganalyzer.

The results show that the proposed model outperformed the existing schemes. A similar observation can be made from Table 3.4, where it has been shown that the predicted cover through the proposed CNN is more accurate than the cover predicted by fixed HPF with respect to the PSNR and SSIM. In Table 3.6, the proposed model steganalytic detection error is tabulated for all the three scenarios on S-UNIWARD, HUGO, and WOW, respectively. From Table 3.6, it can be observed that the proposed model performs better for scenario-2, which is intuitive since the dataset size is double in scenario-2. Unlike Scenario-1 and 2, Scenario-3 is trained on a mix of all the embedding payloads of S-UNIWARD, HUGO, and WOW (2500 image pairs each). Table 3.7 shows the results when detection performance of the proposed scheme is compared to ReST-NET [16] against S-UNIWARD. It is clear that the proposed scheme performs better than ReST-NET [16]. The results presented in Table 3.5 - 3.7 show that the proposed scheme performs better in terms of detection accuracy over state-of-the-art embedding schemes [5, 6, 15] on different payloads.

Table 3.6: *Steganalytic detection error comparison of the proposed scheme under all three scenarios against the S-UNIWARD, HUGO, and WOW embedding scheme. The best result is shown in green, and the second-best result is shown in blue colour.*

Embedding scheme	Proposed scheme	Dataset	0.2 bpp	0.3 bpp	0.4 bpp	0.5 bpp
S-UNIWARD	Scenario-1	BOSSBase	0.191	0.1206	0.0764	0.0532
	Scenario-2	BOSSBase	0.1404	0.0908	0.0588	0.0484
		BOWS2	0.1298	0.0809	0.047	0.0388
	Scenario-3	BOSSBase	0.1685	0.103	0.0771	0.0664
		BOWS2	0.1744	0.0903	0.0645	0.0521
HUGO	Scenario-1	BOSSBase	0.2334	0.1187	0.0842	0.0542
	Scenario-2	BOSSBase	0.1725	0.1104	0.0602	0.0455
		BOWS2	0.164	0.0991	0.0505	0.0351
	Scenario-3	BOSSBase	0.1895	0.1195	0.0896	0.0745
		BOWS2	0.2047	0.1124	0.0779	0.0622
WOW	Scenario-1	BOSSBase	0.2024	0.1130	0.0702	0.0628
	Scenario-2	BOSSBase	0.1716	0.1070	0.0681	0.0513
		BOWS2	0.1599	0.0914	0.0538	0.0440
	Scenario-3	BOSSBase	0.2012	0.1243	0.0926	0.0752
		BOWS2	0.2080	0.1127	0.0776	0.0617

Table 3.7: *Steganalytic detection error comparison of the proposed scheme with the ReST-NET [16] against the S-UNIWARD scheme 0.1 and 0.4 bpp.*

Payload (bpp)	S-UNIWARD	
	Proposed	ReST-NET
0.1	**0.3287**	0.3977
0.4	**0.0588**	0.1427

3.2.3 Implementation Setup

The proposed scheme is implemented using the TFLearn[1] version v0.3 framework, which is compatible with Tensorflow [97] version 1.0 and later versions. The training procedure is carried out on a standard workstation with the NVIDIA Tesla P100 with 16 GB GPU memory and CPU: Xeon E5-2620 with 64 GB memory and 32 CPU cores. The SRNet [4] is also trained on the same machine using Tensorflow [97] to compare with the proposed scheme.

[1] http://tflearn.org/

3.2.4 Summary

In this first contributory chapter, two novel steganalysis schemes have been proposed. The first scheme described in Section 3.1, named *Selective-Signal-Removal* (SSR), is proposed to attack the CMD steganography algorithm. The strategy forces the feature extraction from the regions of the images that are influenced by embedding rather than the whole image. The scheme follows the intuition that the CMD forms the cluster of pixels that are modified in the same direction. The SSR scheme proceeds by determining a suitable threshold value using the DCT of images. Empirically, it is found that the CMD embedding falls in the regions that are marked as above the determined threshold. Therefore, a heuristic function with an assignment algorithm is used to segment the image into relevant (above threshold) and irrelevant super-pixels. The assignment algorithm assigns 1 to the relevant super-pixels and 0 to irrelevant ones. Consequently, a bitmap image is formed. The Hadamard product is performed on the bitmap and the original image to build a new dataset. This new dataset is used to train the SRM with Ensemble Classifier. The experimental results show that the proposed scheme mostly nullified the effect of CMD wrapping and hence increase the classification accuracy by approx 5% in steganalytic classification.

The second scheme discussed in Section 3.2, a deep learning-based steganalyzer has been proposed where the predicted noise residuals are used to train the deep classifier. One of the main contributions of this work is that this noise residual is predicted using a neural network rather than through a conventional high-pass filter, as it is assumed that stego noise may not always present in high-frequency components of a stego image. It has been experimentally shown that such image denoising is more accurate than the conventional fixed high-pass filter-based denoising. Further, it has also been experimentally shown that the classifiers trained using noise residual computed by DCNN perform better for steganalytic detection. A comprehensive set of experiments reveal that the detection performance of the proposed scheme outperformed state-of-the-art steganalyzers.

In the next chapter, the use of heterogeneous context size is explored by designing different multi contextual frameworks for steganalysis.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Chapter 4

Steganalysis: The Role of Hetrogeneous Context Size

The success of the deep-learning steganalytic detectors depends on their network architecture design. Early detectors design started with shallow architecture, and later it is improved by increasing depth, employing batch normalization, removing max-pooling, etc. In this chapter, two different deep-learning-based steganalytic detectors are presented. The first model uses densely connected blocks with progressively increasing context size in each block. The second model uses different context sizes in each block to design a novel steganalytic detector. In the next section, the first model is presented.

4.1 Densely Connected Convnets

It has been observed from the literature presented in Section 1.4.2.2 that the existing CNN-based steganalysis methods: (i) Sharply increase the feature space by using a sequence of kernels in subsequent layers. (ii) Fixed-sized kernels are used, which may not be much expressive in learning the stego features since the stego signal is weak and sparse. Choosing kernel size is critical as lower sized kernel may fail to model noise (embedding) precisely, while a large kernel may lead to overfitting. (iii) Use a fully connected layer at the end for classification.

The use of fully connected layers imposes a constraint that the training and testing must be of the same spatial dimension. In order to use images of different sizes, the images must be resized before testing due to the restriction mentioned above. However, resizing may lead to the loss of stego signals, conceptually similar to pooling. Considering the above drawbacks of the existing schemes, the main contributions of the proposed method are as follows:

- A densely connected convolutional network without pooling layers is proposed, which progressively captures the steganalytic features at different scales.
- The proposed model does not include any fully connected layer, which allows the model to be tested on any image size regardless of the size of images used for training.

The proposed model is trained and tested on BOSSBase 1.0 [85] dataset, and the detection performance is compared with SRM [14], SPAM [13] against S-UNIWARD [15], HUGO [6], WOW [5], and HILL [18]. The proposed method's performance is also compared with recent work [19] on WOW and S-UNIWARD embeddings.

4.1.1 Proposed Work

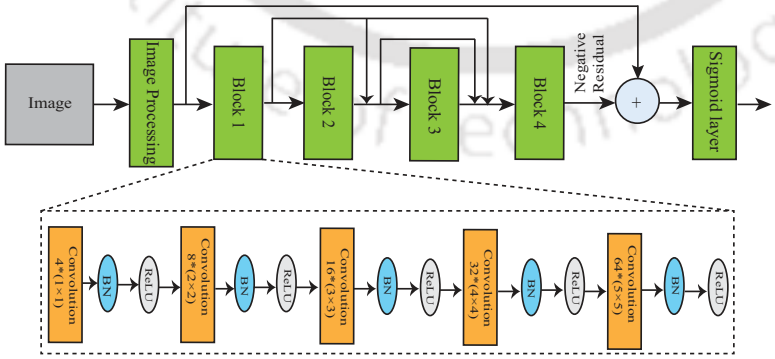


Figure 4.1: *The proposed model architecture. The architecture of each block is similar; one of the blocks (Block 1) is also shown in dotted box. Block consists of* $4 \times (Conv \rightarrow BN \rightarrow ReLu)$ *with sizes indicated for each convolution block.*

This section presents the proposed scheme for *targeted* steganalysis. The proposed model is inspired by DenseNet [68]. The model architecture of the proposed method is shown in Figure 4.1. The proposed model comprises an image processing layer followed by four densely connected convolution blocks and a sigmoid layer for classification. Since the steganalytic classifiers are trained on the noise residual instead of the image components, a fixed high-pass filter (HPF) given in eq. (1.5) has been used in the image processing layer.

The kernel of the image processing layer is kept fixed and is not updated while training. The noise residual extracted from the image processing layer is used as input to the subsequent dense blocks. The densely connected blocks are used to avoid vanishing gradient problems and help extract stego features from a weak and sparse embedding noise. Each block is connected to all its subsequent blocks. Consequently, all the blocks receive the feature map from all their preceding blocks. Each block comprises five convolutional layers. The

Table 4.1: *Details of the layers in each dense block*

layer #	input feature size	# of filters	filter size	Output feature size
1	$1 \times 512 \times 512$	4	1×1	$4 \times 512 \times 512$
2	$4 \times 512 \times 512$	8	2×2	$8 \times 512 \times 512$
3	$8 \times 512 \times 512$	16	3×3	$16 \times 512 \times 512$
4	$16 \times 512 \times 512$	32	4×4	$32 \times 512 \times 512$
5	$32 \times 512 \times 512$	64	5×5	$64 \times 512 \times 512$

details of the layers used in each block are given in Table 4.1. All the blocks have the same configuration except for the last block (Block 4), where the output feature size is $1 \times 512 \times 512$. Convolutional layers in each block are followed by the *Batch Normalization* (BN) [58] for faster convergence and the ReLU [98] activation. The pooling layer has not been used since the use of pooling may result in loss of the stego noise. The number of convolutional filters progressively increases as $4, 8, 16, 32$, and 64, and the kernel size also increases gradually from 1×1 to 5×5 as each block slowly increases the scope of the convolution operator. The different sized kernels help to learn the features at different scales, thereby avoiding the loss of the stego signal and capturing more prominent features.

The densely connected blocks' output is a negative residual map that is pixel-wise added to noise residual extracted by the image processing layer to boost the noise components. The resulting output is used as input to the classification layer (sigmoid layer). The classification layer determines whether the input image is a stego or cover image using the mean sigmoid over entire pixels. The network is trained by minimizing the cross-entropy loss function.

4.1.2 Implementation Details and Results

4.1.2.1 Experimental Setup

The experiments are carried out on the BOSSBase v1.0 dataset [85]. The dataset consists of 10,000 cover images of size 512×512. The steganographic embedding algorithms[1] S-UNIWARD, HUGO, WOW, and HILL, are used to obtain stego images. Further, 10000 cover-stego pairs of images are divided into training: 5000, validation: 1000, and testing: 4000 cover-stego pairs. To compare the performance, SRM and SPAM are also implemented with the same split as the proposed model. The proposed model is implemented using Pytorch [99] on a standard workstation having NVIDIA Quadro M-4000 GPU (8GB) for 90 epochs. The learning rate is initially set to 0.001 and decays by a factor of 10 every 30 epochs. The batch size is empirically kept as 8 (4 cover and 4 stego). Adam Optimizer [100] is used to optimize the network parameters when training.

4.1.2.2 Results

The quantitative results are given in Table 4.2 when compared to the proposed model is compared with SRM with EC [17] and SPAM with EC against S-UNIWARD, HUGO, WOW, and HILL embedding with different payloads. The results are measured in terms of percentage (%) classification accuracy. The best result is shown in boldface. A series of graphs are also given in Figure 4.2 for a visual presentation where the proposed scheme is shown in red color, SRM with EC is

[1]Steganographic algorithms, feature extractors such as SRM, SPAM and Ensemble classifier can be found at: http://dde.binghamton.edu/download/

Table 4.2: *Steganalytic classification accuracy (in %) of the proposed scheme is compared to SRM [14] with Emsemble classifier [17] and SPAM [13] with Ensemble classifier against S-UNIWARD [15], HUGO [6], WOW [5] and HILL [18].*

Scheme	Payload(bpp)	Proposed scheme	SRM with EC	SPAM with EC
S-UNIWARD	0.1	**66.50%**	59.05%	54.24%
	0.2	**68.50%**	62.17%	58.92%
	0.3	**70.75%**	65.80%	63.44%
	0.4	**75.25%**	73.70%	67.51%
HUGO	0.1	**63.50%**	60.03%	52.36%
	0.2	**70.25%**	67.98%	56.00%
	0.3	74.00%	**74.46%**	60.03%
	0.4	77.25%	**78.30%**	63.98%
WOW	0.1	**67.25%**	60.97%	52.46%
	0.2	**70.75%**	65.77%	55.82%
	0.3	**74.00%**	69.26%	58.98%
	0.4	**76.50%**	75.12%	62.33%
HILL	0.1	**61.50%**	55.45%	52.28%
	0.2	**65.75%**	61.37%	55.26%
	0.3	**69.50%**	67.11%	58.41%
	0.4	**75.00%**	72.58%	61.37%

Table 4.3: *Comparison of the proposed scheme with Tian and Li [19] in terms of steganalytic classification accuracy (in %) against WOW [5] and S-UNIWARD [15].*

Payload	WOW		S-Uniward	
bpp	Proposed scheme	Tian and Li [19]	Proposed	Tian and Li [19]
0.1	67.25%	**67.90%**	**66.50%**	65.10%
0.3	**74.00%**	69.00%	**70.75%**	67.20%
0.4	**76.50%**	71.4%	**75.25%**	69.80%

shown in black color and SPAM with EC is shown in blue. It can be observed from the results that the proposed scheme outperformed SRM [14] as well as SPAM for most of the steganographic algorithms. The proposed scheme's steganalytic performance is also compared with *Tian and Li's* recent work [19], which has the same experimental setup as the proposed method. The results are given in Table 4.3. The proposed scheme has comparable performance against WOW [5] on 0.1 bpp, and for the rest of steganographic embedding and payloads, the proposed method outperformed Tian and Li [19].

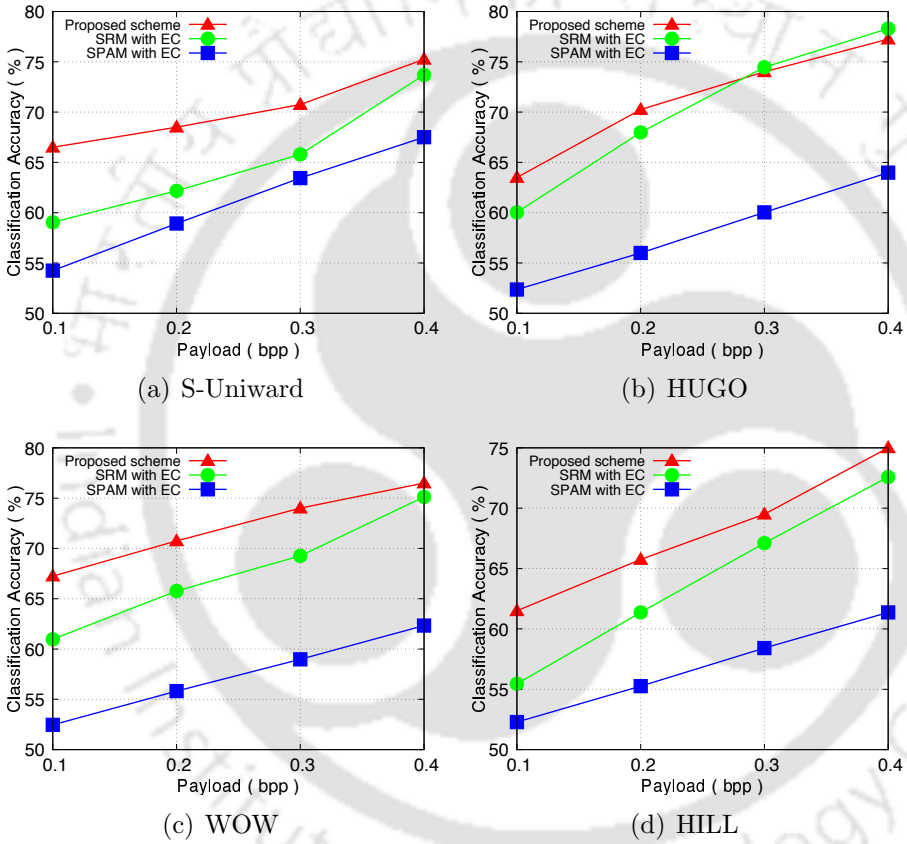


Figure 4.2: *Steganalytic performance comparison of the proposed scheme (red) with SRM with EC (green) and SPAM with EC (blue) against S-UNIWARD, HUGO, WOW, and HILL steganography on embedding rates - {0.1, 0.2, 0.3, 0.4} bpp.*

4.2 Multicontextual Design of CNN for Steganalysis

As it is observed from the previous work, recent steganographic embedding does not always restrict their embedding in the high-frequency zone; instead, they distribute it as per embedding policy. Thus learning the embedding zone is a challenging task. This section introduces a deep neural network that can extract multiple contextual features from noise residuals. Unlike conventional approaches, the proposed model extracts noise residual using learned denoising kernels to boost the signal-to-noise ratio. After preprocessing, the sparse noise residuals are fed to a novel *Multi-Contextual Convolutional Neural Network* (M-CNet) using a heterogeneous context size to learn the fine-grained noise residuals. The model performance is further improved by incorporating the *Self-Attention* module to focus on the areas prone to steganalytic embedding.

The design of steganalytic detectors can be roughly divided into two types - the first type is the methods that use some preprocessing element to compute noise residual and then perform detection utilizing the residuals. The second type is an end-to-end network which does not require any explicit preprocessing elements. The following observations are made from the existing literature given in Section 1.4.2:

1. A majority of the methods [2, 39, 60, 63, 101] use fixed high-pass filters in preprocessing stage for noise residual computation.

2. The computed noise residuals are very low-amplitude signals that require a robust detector to be learned.

3. The modern steganography algorithms distribute the embedding in the less predictable regions of the carrier image. Therefore, a mechanism is needed for the detectors to focus on these parts as well as the usual highly probable image regions.

With the above motivation, our contributions are four-fold:

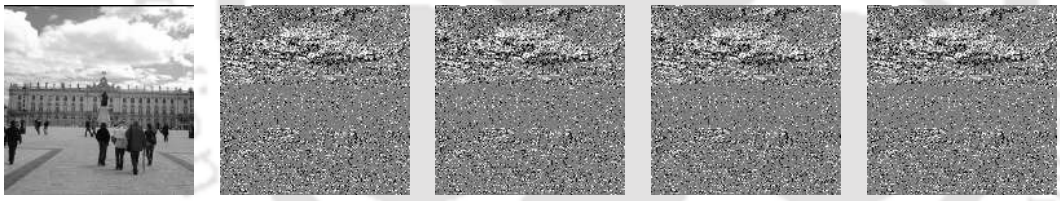
1. A set of thirty filters is learned using a CNN model to replace the fixed high-pass filters used in preprocessing steps for residual computation from images [102].

2. A multi-context design of a steganalyzer is devised to learn the low-amplitude and sparse noise residuals.

3. The proposed model uses a Self-Attention [103] mechanism to focus on areas that are likely to be affected by embeddings.

4. Additionally, an ablation study is presented at the end for justification of the proposed architecture.

4.2.1 Proposed Method



(a) Cover (b) WOW (c) S-UNIWARD (d) HILL (e) MiPOD

Figure 4.3: *Stego noise embedded by different steganographic algorithms - (b) WOW, (c) S-UNIWARD, and (d) HILL (e) MiPOD with payload = 0.4 bpp*

The name of the proposed model is *M-CNet*, "Multi-contextual network for steganalysis." In this model, we have used convolutions with different kernel sizes to get the responses having different contexts, hence the name "multi-context." We believe that steganographic noise is not equally visible with a particular kernel. Instead, it (noise) can be traced more prominently with suitable kernel size depending on the local image statistics (context). The proposed model can be roughly divided into two modules. The first module is an extension of our previous work [102], and the second module is a multi-context CNN model.

4.2.1.1 Rationale

In principle, the modern content-adaptive steganography schemes hide most steganographic embedding in the noisy or texture region and relatively less in the smooth regions. The same observation can be made through Figure 4.3, where (a) shows the cover image and (b)-(e) depicts the steganographic embedding done by WOW [5], S-UNIWARD [15], HILL [18], and MiPOD [104], respectively. Noticeably, Figure 4.3 (b)-(e) shows that most steganographic embeddings are clustered towards the high-texture regions and are sparsely distributed across the smooth areas. This observation drives the steganalysis schemes to focus on the high-textured region to learn the features that may lead to better detection. Steganalysis methods [2, 13, 14, 39, 60, 101] use different high-pass filters to suppress the image content and expose the noise content of an image. This allows the detectors to train on the noise domain rather than the image domain. However, this may not always be true. There are some recent embedding schemes that embed not only in the high-frequency zone but also in smoother areas [105, 106]. For these cases, high-frequency-based filters may not be very useful.

With this motivation, we propose a denoiser subnetwork to compute the noise residual from a given image. For simplicity of reference, we call the denoiser subnetwork ϕ_{DN} subnetwork.

4.2.1.2 Denoiser Subnetwork ϕ_{DN}

The denoiser (ϕ_{DN}) subnetwork extends one of our previous works [102], where the cover image is predicted using a single-layered CNN from a stego image. Later, the noise residual is computed as the pixel-wise difference between the stego and the predicted cover images to train a classifier. We refer to the subnetwork mentioned above as "denoiser" due to its post-processing adaptability of predicting the stego noise residual. However, in this work, we train the ϕ_{DN} subnetwork to estimate the noise residual, bypassing the post-processing step directly. Furthermore, instead of utilizing 16 filters as in [102], we propose to use

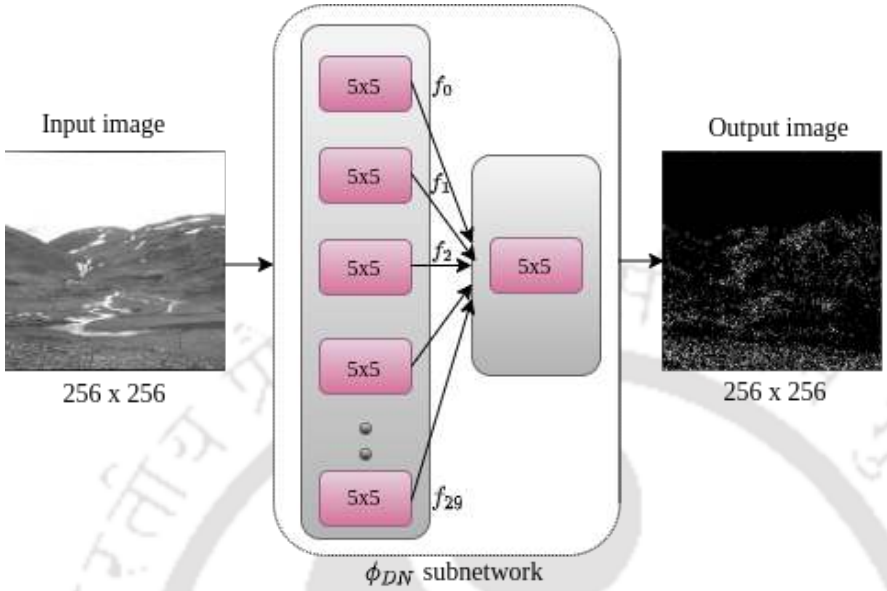


Figure 4.4: *Architecture of the Denoiser subnetwork (ϕ_{DN}).*

30 filters to extract the rich set of features that may capture the stego embeddings in both low and high-frequency zones. We have shown that such empirical changes in the ϕ_{DN} subnetwork's engineering may lead to a notable reduction in the detection error probability (see Section 4.2.4.1). A graphical overview of the ϕ_{DN} subnetwork has been presented in Figure 4.4. Architecturally, the denoiser subnetwork consists of two convolution layers, with 30 and 1 filters, respectively. To formally define the regime of operations of the ϕ_{DN} subnetwork, let X, Y denote the cover and stego images, respectively. Consequently, the target cover and stego noise residuals can be written as $\mathcal{N}^c = |X - X|$ and $\mathcal{N}^s = |Y - X|$, respectively. Given an input image X or Y, the proposed ϕ_{DN} subnetwork aims to estimate the corresponding $\mathcal{N}^c$ or $\mathcal{N}^s$ noise residual. The thirty kernels of the first convolution layer in the ϕ_{DN} subnetwork are initialized with SRM filters. It has been observed that such an initialization leads to faster convergence and a significant boost in detection accuracy. It should be mentioned that the pixel-wise formulation of noise residual may capture stego embeddings in both low and high-frequency zones of an image. However, unlike [102], in this work,

we utilize the inherent knowledge of the ϕ_{DN} subnetwork to train the proposed multi-contextual classifier. Precisely, instead of the predicted noise residual, we input the intermediate feature maps $f_0, f_1, ...f_{29}$ generated by the 30 filters of the first layer in ϕ_{DN} to the M-CNet. By way of analysis, the intermediate feature representations of the final noise residual may have a variety of finer details (see Figure 4.5) that can be better leveraged by the multi-contextual filters of the proposed M-CNet. To the best of our knowledge, this is the first work that distills and incorporates the inherent information of the denoiser subnetwork in terms of intermediate feature representations and optimally trains the classifier.

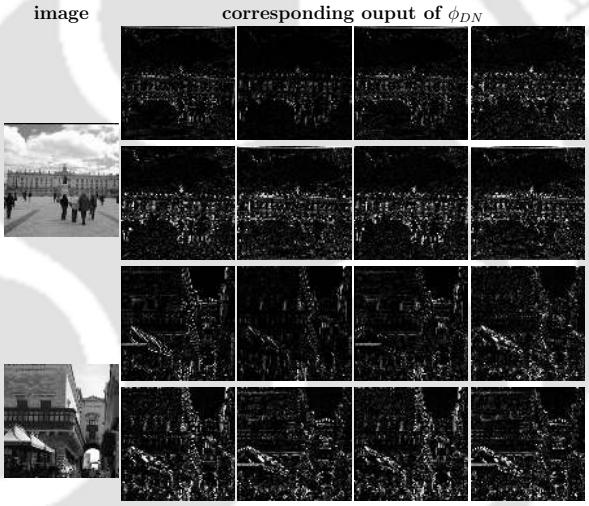


Figure 4.5: *Sample output of ϕ_{DN} subnet: The first column represents input image, and columns 2 to 5 show the noise residual maps when preprocess using kernels of ϕ_{DN} subnet.*

4.2.1.3 Multi-context subnetwork

The purpose of the multi-context subnetwork is to learn the discriminative features that may not be learned using the fixed-size kernels due to the non-uniform sparsity of the stego noise in an image. The multi-context subnetwork is combined with the preprocessing element ϕ_{DN} to form the M-CNet. As shown in Figure 4.6, the multi-context subnetwork comprises six convolution blocks, followed by a *Self-Attention* layer [103], followed by a Global Average Pooling (GAP)

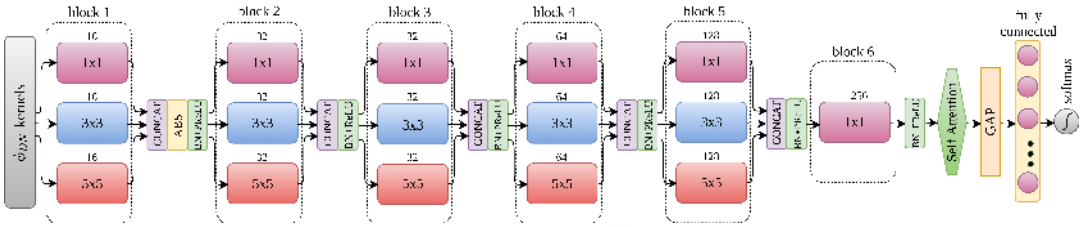


Figure 4.6: *Architecture diagram of the M-CNet.*

layer, and a fully-connected layer followed by the softmax layer. Each convolution block (*block1* to *block5*) is equipped with three different kernel sizes - 1×1, 3×3, and 5×5 to learn the noise residual at different contextual orientations. The learned features are then concatenated by using a *CONCAT* layer before feeding to the next layer. After concatenation, the features are normalized using the *Batch Normalization* (BN) [107] for the faster convergence of the network, followed by a Parametric ReLU (*PReLU*) non-linearity [108] before feeding to the next layer. The *ABS* layer is applied to the concatenated features, which offers better learning of noise residual by preserving the negative as well as positive features [39]. The *block6* is kept with a filter of size 1×1, which reduces modeling strength by keeping the salient features. The stego noise is fine-grained and sparse features scattered across the entire residual map. Therefore, a mechanism is needed to learn these noises and focus on the areas that are likely to be most affected by embeddings. With this motivation, a *Self-Attention* layer is used to enable the network to learn the features from the regions that are most likely to be affected by the steganographic embeddings. The employed Self-Attention mechanism adaptively refine the learned noise residual (output of the PReLU after *block6*) by suppressing the irrelevant multi-contextual features. After the Self-Attention layer, the Global Average Pooling layer is used to reduce the dimensionality of the features by retaining only essential features. Following the *GAP* layer, a fully connected network layer followed by the *softmax* is used for binary classification using learned features. In order to come up with the final architecture, an extensive set of experiments have been conducted; the details are

presented in Section 4.2.4.

4.2.2 Experimetal Study

This section presents the dataset details, steganography algorithms, and the evaluation metric used to train and evaluate the models.

4.2.2.1 Dataset

The performance of the proposed M-CNet is compared with SCA-Yenet [60] and SRNet [4]. Therefore, the same dataset configuration has been used for training, testing, and validation, with some minor changes. The entire dataset is consists of the union of BOSSbasee 1.01 [85] and BOWS-2 [86]. Both of these datasets contain 10,000 grayscale images of size 512×512. Each of these images is first resized to 256×256 using the *imresize* function of Matlab. The training set is comprised of 4,000 randomly selected images from the BOSSbase1.01 dataset and the entire BOWS2 images (10,000). The validation set consists of randomly selected 1,000 images from the remaining BOSSbase dataset and the rest 5,000 images are used for testing. Therefore, the training set contains $2 \times 14,000$ images (14,000 cover and 14,000 stego), the validation set includes $2 \times 1,000$ images, and the test set contains $2 \times 5,000$ images. Further, $2 \times 2,000$ pairs are randomly selected from the training set to train and validate ($2 \times 1,600$ for training, and 2×400 for validation) the ϕ_{DN} subnetwork. The stego image dataset is generated with S-UNIWARD [15], WOW [5], HILL [18], and MiPOD [104] steganography algorithms[1] using random keys.

All the models reported in this work are trained on the same dataset splits, as reported in Section 4.2.2.1. The models are implemented using PyTorch [99] on Nvidia V-100 with 32 GB GPU memory. Each epoch of M-CNet training takes $\approx$ 22 minutes, approximately six days to train for 400 epochs.

[1]The code for steganography algorithms are downloaded from http://dde.binghamton.edu/download/stego_algorithms/

4.2.2.2 Training

The proposed model is trained for the detection of spatial domain steganography schemes.

Training ϕ_{DN} subnetwork: The ϕ_{DN} subnetwork is trained using $2 \times 1,600$ images and validated with 2×400 images. These images are not overlapped with any of the training, validation, and testing dataset of the proposed M-CNet model. The ϕ_{DN} is trained for 100 epochs, with a mini-batch of 20 images (10 cover and 10 stego), using the Adamax [109] optimizer. The learning rate is initialized to 10^{-3} and decayed every 25 epochs by a factor of 10^{-1}. Each mini-batch images are subject to data augmentation, rotation by 90°, horizontal and vertical flipping, with a probability of 0.4. The ϕ_{DN} subnetwork is trained by minimizing the *mean-squared* loss ($\mathcal{L}_2$).

$$\mathcal{L}_2 = \|\hat{Y} - Y\|_2^2, \tag{4.1}$$

where Y and $\hat{Y}$ denote the input and the image estimated by ϕ_{DN} subnetwork, respectively. After the training, the model with the minimum validation error is chosen for the preprocessing of the proposed M-CNet model.

Training M-CNet: The training of M-CNet is carried out on $2 \times 12,000$ training images. Adamax [109] optimizer is used with a minibatch of 20 images (10 cover and 10 stego) for 400 epochs. The learning rate is initialized to 10^{-3} and decayed every 40 epochs by a factor of 10^{-1}. All the weights of convolutional kernels are initialized with *Xavier initializer* [110], and biases are initialized with 0. The weights of fully connected neurons are initialized with Gaussian distribution with $\mu = 0.0$ and $\sigma = 0.01$. Each mini-batch of training is subject to data augmentation by horizontal and vertical flipping and rotation by 90° with a probability of 0.4. The M-CNet is trained by minimizing the binary cross-entropy

loss $\mathcal{L}_{BCE}$.

$$\mathcal{L}_{BCE} = -\frac{1}{N}\sum_{i=1}^{N}(y_i.log\hat{y}_i + (1-y_i).log(1-\hat{y}_i)), \tag{4.2}$$

where N denotes the number of training examples, y and $\hat{y}$ represent the true label and the predicted label, respectively.

In this work, we have also given the *Receiver Operating Characteristics* (ROC), *Area Under Curve* (AUC), and the *Weighted Area Under Curve* (WAUC)[1] as an alternative metric of evaluation.

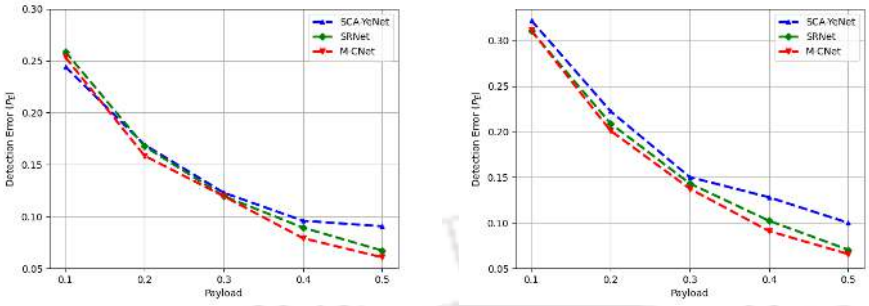
The results presented in Section 4.2.3 are reported for a random 50-50 split of the BOSSBase1.01 dataset. However, to evaluate the detection performance across different splits, we trained the proposed M-CNet model on five different random 50-50 splits of BOSSBase, keeping BOWS2 fixed in training set on WOW with payload 0.4 bpp. As a result, the standard deviation of detection error (P_E) is obtained ≈ 0.00359 for these five splits.

4.2.3 Results and Discussion

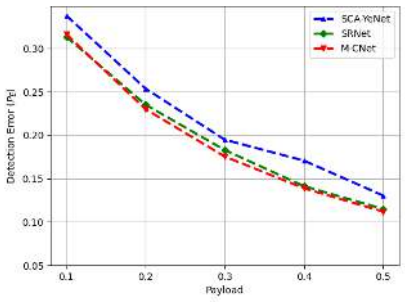
In this section, the results of the M-CNet are presented and compared with the state-of-the-art detectors.

The detection performance of the proposed M-CNet model is compared with the prior arts, namely, YeNet [60] and SRNet [4]. YeNet and SRNet are two spatial domain steganalytic detectors, which achieved state-of-the-art detection accuracy in the spatial domain. Both the methods are based on different training mechanisms. YeNet is based on the training with preprocessing filters, whereas SRNet is based on end-to-end training without preprocessing. The detection performance of these methods is compared with the proposed M-CNet while detecting the spatial domain steganography schemes, such as WOW [5], S-UNIWARD [15], and HILL [18]. For a fair comparison with the proposed M-CNet, YeNet and SRNet are implemented with the experimental setup stated in the respective papers.

[1]code to compute WAUC is downloaded from: https://www.kaggle.com/c/alaska2-image-steganalysis/overview/evaluation



(a) WOW (b) S-UNIWARD



(c) HILL

Figure 4.7: *Comparison among SCA-YeNet, SRNet, and M-CNet in terms of detection error probability P_E on BOSSBase on (a) WOW, (b) S-UNIWARD, and (c) HILL steganography.*

The detection error probability P_E (defined in eq. (2.3)) of the steganalyzers is recorded in Table 4.4. The ROC curve for M-CNet is shown in Figure 4.8, and AUC/WAUC scores are given in Table 4.5.

Curriculum Learning for training with lower payloads: IIt is a well-known fact that when a detector is trained on a steganography algorithm with a low payload, it performs worse, and sometimes it may not converge at all while training [59]. The solution to this kind of problem has been found using *curriculum and transfer learning* [111, 112]. Some of the recent steganalysis schemes [4, 60] used curriculum learning for training the model with the lower payloads. Following this trend, we trained the proposed M-CNet model with a

Table 4.4: *Detection error probability P_E for M-CNet, YeNet, and SRNet. The best results are indicated in* ***bold face****.*

Embedding	Detector	Payload (bpp)				
		0.1	0.2	0.3	0.4	0.5
WOW	SCA-YeNet	**0.2442**	0.1691	0.1229	0.0959	0.0906
	SRNet	0.2587	0.1676	0.1197	0.0893	0.0672
	M-CNet	0.2555	**0.1649**	**0.1115**	**0.0759**	**0.0563**
S-UNI	SCA-YeNet	0.3220	0.2224	0.1502	0.1281	0.1000
	SRNet	0.3104	0.2090	0.1432	0.1023	0.0705
	M-CNet	**0.3100**	**0.2010**	**0.1375**	**0.0910**	**0.0658**
HILL	SCA-YeNet	0.3380	0.2538	0.1949	0.1708	0.1305
	SRNet	**0.3134**	0.2353	0.1830	0.1414	0.1151
	M-CNet	0.3165	**0.2301**	**0.1755**	**0.1389**	**0.1120**

higher payload (0.4 bpp) and then transferred its weights to train it for lower payloads (0.3 bpp) using a very small learning rate (10^{-7}), reducing every 100 epoch by a factor of 10^{-1}. The proposed M-CNet model using the *curriculum learning* is fine-tuned for 200 epochs. The proposed M-CNet model with *curriculum learning* is evaluated for epochs between 51 to 200. The model with the best validation P_E is evaluated, and results are presented in Table 4.4.

The results presented in Table 4.4 and Figure 4.7 show that the proposed M-CNet performed well when it is tested on WOW and S-UNIWARD while it attains the performance comparable to SRNet [4] on HILL embedding. It should also be noted that better detection performance is obtained for the embedding algorithms with higher payloads, such as 0.3 - 0.5 bpp. Further, the ROC curve presented in Figure 4.8 and AUC and WAUC (see Table 4.5) is given as the alternate measure of detection performance. The ROC and AUC also depict that the proposed model performed better for the embedding algorithms WOW and S-UNIWARD than HILL embedding. While the method such as SRNet design consideration is based on residual connections and end-to-end learning,

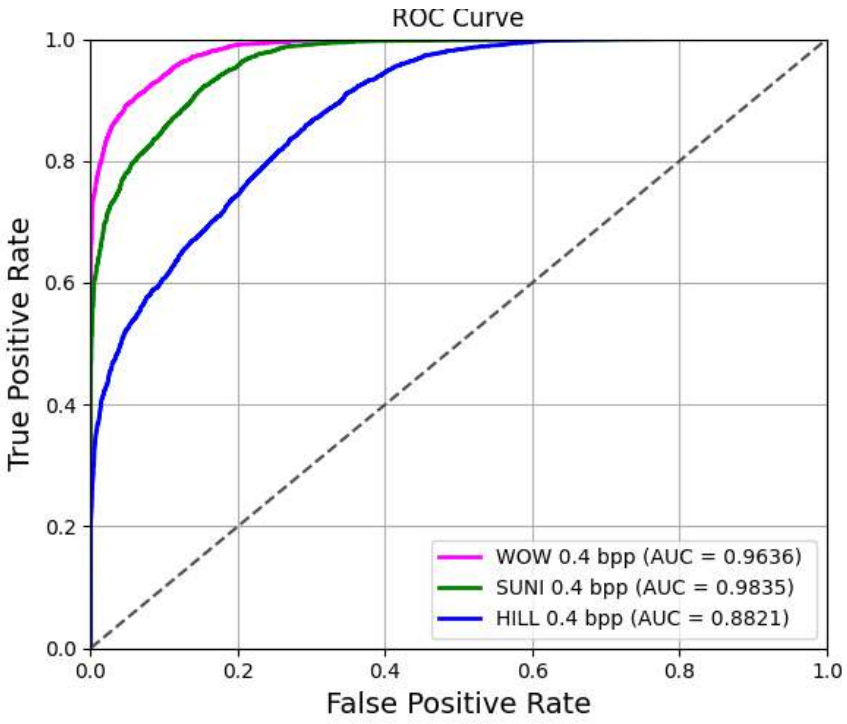


Figure 4.8: *ROC curve of M-CNet when tested on WOW, S-UNIWARD, and HILL steganography with 0.4 bpp.*

the proposed M-CNet is explicitly designed to learn the sparse and low-amplitude noise residual extracted by its preprocessing network(ϕ_{DN}). The experimental results show that the proposed model outperformed the prior arts. The detection efficacy of the proposed M-CNet model might be due to the multi-contextual design, which may have helped in learning the sparse and low-amplitude noise residual spread over both low and high-frequency regions. Later, the adopted *self-attention* mechanism might have helped in focusing on those regions more by suppressing the irrelevant regions.

Further, we evaluated the proposed M-CNet model with different architectural configurations. The details of the experiments are presented in Section 4.2.4.

Table 4.5: *AUC and WAUC scores of the M-CNet when detecting different types of steganography at differnt payloads.*

Embedding	Metric	Payload (bpp)			
		0.2	0.3	0.4	0.5
WOW	AUC	0.9324	0.9686	0.9835	0.9885
	WAUC	0.9517	0.9776	0.9883	0.9917
SUNI	AUC	0.8848	0.9143	0.9636	0.9830
	WAUC	0.9170	0.9387	0.9740	0.9879
HILL	AUC	0.8066	0.8592	0.8821	0.9066
	WAUC	0.8425	0.9343	0.9502	0.9685

4.2.4 Ablation Study

This section presents a brief ablation study to assess the proposed model with different configurations and various scenarios. Unless explicitly specified, all the experiments presented in this section are performed on WOW steganography with a payload of 0.5 bpp.

4.2.4.1 Configurations of Kernels in the ϕ_{DN}

An obvious question one may ask is, how do the different choices of ϕ_{DN} affect the performance of the proposed M-CNet model? Our goal with the ϕ_{DN} is to learn the kernels that can improve the noise residual computation. Therefore, we kept our search confined to a single layer. Table 4.6 presents the different configurations of filters explored for the ϕ_{DN}. The first column represents the number of filters (n), the second and the third column represent the filter size (s) used and the corresponding detection error probability P_E, respectively. The best P_E is achieved in this experiment for $n = 30$ and 32, and $s = 3 \times 3$. These results are obtained when all the kernels are initialized with the *Kaiming initialization* [108]. We also investigated the initialization of the kernels using SRM, which has kernel sizes of 1×1 to 5×5. The smaller size SRM kernels are padded to zeroes to match the dimension of 5×5. In order to initialize with the SRM kernels, we kept the no. of filters $n = 30$ and kernel size $s = 5 \times 5$. The details of initialization are presented in Section 4.2.4.2. We also explored

a few smaller architectures with a few layers for the ϕ_{DN} but could not get any promising results.

Table 4.6: *Detection error probability (P_E) with different filter size and no. of filters in ϕ_{DN}.*

# filters (n)	filter size (s)	P_E
16	3×3	0.0820
	5×5	0.0873
30	3×3	**0.0800**
	5×5	0.0810
32	3×3	0.0811
	5×5	0.0830
64	3×3	0.0895
	5×5	0.0937

4.2.4.2 Kaiming vs. SRM vs. Gabor initialization

It has been widely established that the learning-based models may achieve poor convergence when initialized with the random weights derived from the Gaussian distribution [110]. To overcome this drawback, we experimented by initializing the kernels of the ϕ_{DN} module with different filters, namely, *Kaiming* [108], SRM filters [14], and *Gabor* filters [62]. For our experiment, 30 *Gabor* filter are obtained with following parameters: Scale $\sigma \in \{0.5, 1\}$, Wavelength of the cosine function $\lambda = \sigma/0.56$, Spatial aspect ratio $\gamma = 0.5$, and fifteen orientations $\theta \in [0, 14\pi/15]$. The experimental results are tabulated in Table 4.8. The SRM, and *Gabor* Kernels are first padded with zeroes to match the dimension to 5×5 before the initialization. The ϕ_{DN} converged faster while training when initialized with SRM kernels. We also evaluated the performance of the proposed M-CNet model with these initializations; results are shown in Table 4.7. The results show that the proposed M-CNet model also performed better when initialized with *SRM* kernels.

Table 4.7: *Detection error probability when the ϕ_{DN} subnetwork is initialized with different types of kernels.*

Initialization	Kaiming	SRM	Gabor
P_E	0.0810	**0.0563**	0.0723

4.2.4.3 Split vs. end-to-end training of the ϕ_{DN} subnetwork

The training of the proposed M-CNet model is carried out in two phases (we call it split training). In the first phase, the ϕ_{DN} subnetwork is trained. In the second phase, the learned weights of the ϕ_{DN} are used in preprocessing, and the entire network is trained, keeping the learned kernel weights of the ϕ_{DN} fixed. Nevertheless, to investigate the performance of the proposed M-CNet model when trained in the end-to-end fashion, both the networks (ϕ_{DN} and multi-context subnetworks) are clubbed together, keeping the ϕ_{DN} in the front end of the M-CNet. As a result, the error detection probability P_E is found to be 0.0848 for end-to-end trained M-CNet, which is inferior to the performance with the split training (0.0563).

4.2.4.4 Detection performance of the proposed M-CNet when the ϕ_{DN} subnetwork is replaced with handcrafted filters like- SRM, KV, or Gabor

To this end, we trained the model without any preprocessing and with preprocessing using KV, SRM, and Gabor filters [62]. The KV filter is a single 5×5 filter, which has been used in numerous steganalyzers [2, 39] for preprocessing. The SRM [14] filter bank consists of 30 linear and non-linear filters. All these kernels are resized to 5×5 before using with M-CNet. A set of 2D Gabor filters [62] has been used by Li *et al.* [16] in the preprocessing stage of the detector for better extraction of noise residual. The Gabor filters are obtained using the setting stated in Section 4.2.4.2. The best detection error probability is found when the model is trained with preprocessing using ϕ_{DN} kernels. The result shows that the preprocessing elements, which are learned, are comparatively better than the

Table 4.8: *Detection error probability when different kind of filters are used in pre-processing stage of the proposed M-CNet.*

Preprocessing using	P_E
No filter	0.1115
KV filter	0.1079
Gabor filter	0.0959
SRM filters	0.0824
ϕ_{DN} kernels	**0.0563**

fixed ones.

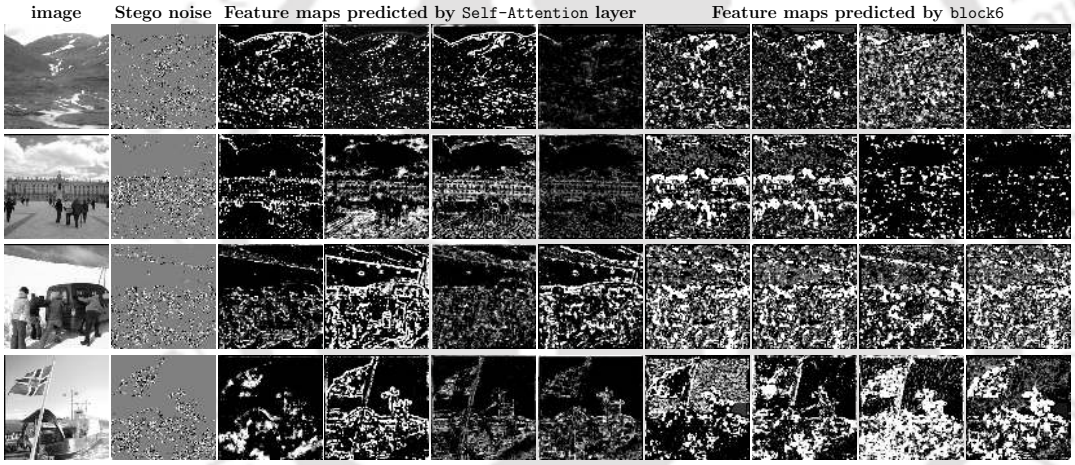


Figure 4.9: *Sample: The first two columns represent an image and corresponding noise residual ($\mathbf{N} = \mathbf{Y} - \mathbf{X}$). The next four columns (col. 3-6) show the 4 feature maps (out of 256) predicted by the* `Self-Attention` *layer of M-CNet, and the col. 7-10 show the 4 feature maps (out of 256) predicted by* `block6` *(M-CNet without* `Self-Attention`*).*

4.2.4.5 Choice of filter configuration in multi-context subnetwork

We carried out experiments with different configurations of the proposed M-CNet, keeping only one type of kernel in each convolutional layer except for the last layer where 1x1 kernel size is used. The results of this experiment are presented in Table 4.9. When all the layers of the proposed M-CNet are equipped with the 1×1 kernel size, the detection error is found to be 0.1319, which might

be due to the smaller context size alone is not suitable for capturing the stego features. The better detection error 0.0968 is obtained for the model with kernel size 3×3. Nevertheless, considering the low amplitude and sparse characteristics of stego noise, any single type of kernel alone may not be suitable to capture such features. Therefore, we further explored the models with multiple context sizes in each layer to capture these features more precisely. The results of this experiment are presented in Table 4.10. We initially experimented with combinations of two different size kernels in each convolution layer, keeping the last layer fixed with only 1x1 kernels. Finally, we experimented with all three different kernel sizes in each layer. The model with this configuration performed better than the former combinations.

Table 4.9: *Detection error probability when only one type of filter size is used in each layer of the multicontext network.*

Layers configuration	P_E
All layers 1×1	0.1319
All layers 3×3	**0.0968**
All layers 5×5	0.1010
First two layers $5 \times 5 \rightarrow$ three layers 3×3	0.1054

Table 4.10: *Detection error probability P_E when multiple filters of different size are used in each layer of the network.*

Layers configuration	P_E
1×1 and 3×3	0.0939
1×1 and 5×5	0.0872
3×3 and 5×5	0.0786
1×1, 3×3, and 5×5	**0.0563**

4.2.4.6 Increasing and Decreasing depth and no. of filters per layer of the proposed M-CNet

An extensive set of experiments have been conducted to develop the final architecture of the M-CNet; some of them are presented here. The first question is, how does the performance changes along with the depth of the network? To this end, we conducted the experiments keeping all the components fixed and varying only the depth (d = no. of blocks) from 2 to 8. Beyond $d = 8$, the model exhausted the available resources for computation. The result presented in Table 4.11 shows that the M-CNet performed best for $d = 6$ and 7. However, we kept the configuration of the M-CNet model to $d = 6$ since it saves the number of trainable parameters and lowers the training time. It can infer from the results that the model with $d = 2$ is very weak to learn the stego embedding, whereas, for $d = 8$, the low-amplitude stego noise may vanish along with the depth of the network.

Table 4.11: *Detection error P_E with variation in depth of M-CNet.*

Depth (d)	2	3	4	5	6	7	8
P_E	0.2239	0.1080	0.0980	0.0729	**0.0563**	0.0610	0.0759

4.2.4.7 Choice of activation

There are various choices for activation function such as *Sigmoid*, *TanH*, *ReLU*, Leaky ReLU ($LReLU$), Parametric ReLU ($PReLU$).

$$Sigmoid(\theta_i) = \frac{1}{1+e^{-\theta_i}}; \quad TanH(\theta_i) = \frac{e^{\theta_i} - e^{-\theta_i}}{e^{\theta_i} + e^{-\theta_i}},$$

$$f(\theta_i) = \begin{cases} \theta_i, & \text{if } \theta_i > 0 \\ \alpha_i.\theta_i, & \text{if } \theta_i \leq 0, \end{cases}$$

where θ_i denote the input to the nonlinear activation on the i^{th} channel. $f(\theta_i)$ denote the $ReLU$ activation when the parameter $\alpha_i = 0$, $LReLU$ when α_i is fixed to a small constant ($\alpha_i = 0.01$), and $PReLU$ when α_i is learned with the model.

The aim of $LReLU$ is to avoid the zero gradients that occur in $ReLU$. But, experiments in [113] showed that replacing the $ReLU$ with the $LReLU$ has a very negligible impact on accuracy. However, $PReLU$ adaptively learns the parameters jointly with the model and also has a considerable impact on the accuracy [108]. For steganalytic detectors, $PReLU$ helps to learn the positive as well negative embedding more precisely by retaining the negative values using learned parameters, which might be lost when $ReLU$ is used. We experimented with *Sigmoid*, *TanH*, $ReLU$, and $PReLU$ activation to compare the performance when different types of activation are used. The P_E for each of these activations is recorded in Table 4.12. Experimentally, the best detection error is observed when $PReLU$ activation is used throughout the network.

Table 4.12: *Detection error P_E with variation in depth of M-CNet.*

Activation	Sigmoid	TanH	TanH→ReLU	ReLU	PReLU
P_E	0.1811	0.0776	0.0670	0.0607	**0.0563**

4.2.4.8 With or without Self-Attention?

To answer this question, we trained the M-CNet without the *Self-Attention* [103] and compared the performance with the M-CNet. The detection error is found to be $P_E = 0.0855$. A set of samples for the model with and without the Self-Attention are presented in Figure 4.9. In Figure 4.9, col. 1 shows a cover image, col. 2 shows the noise embedding ($\mathcal{N} = Y - X$) by WOW steganography with payload 0.5 bpp, the col. 3-6 represent the features predicted by the Self-Attention layer of the M-CNet, and col. 7-10 represent the features predicted by *block 6* when the Self-Attention layer is not used in M-CNet. The samples show that the outputs predicted by the layers provide a variety of features by assigning high values (255) to the highly textured and low values (0) to the smooth regions. Nevertheless, it can be clearly observed that the output maps predicted by Self-Attention focus more on the regions that contain steganographic embeddings. This precise feature learning of Self-Attention caused the increase in detection

accuracy by $\approx 3\%$.

4.2.4.9 Detection performance of the proposed M-CNet for stego-source mismatch

To this end, we trained the proposed M-CNet model on one steganography algorithm and tested it with another with the same payload. The results presented in Table 4.13 show that when the model is trained on the easy to detect algorithm WOW and tested on the hard-to-detect algorithm MiPOD, the model performed worse. Comparatively, it performed better when trained on MiPOD and tested on WOW and several other steganography.

Table 4.13: *Detection error probability P_E of the proposed M-CNet for stego-source mismatch scenario on different embedding with payload 0.4 bpp*

Train\Test	WOW	S-UNI	HILL	MiPOD
WOW	0.0759	0.1479	0.3301	0.2850
S-UNI	0.1082	0.0910	0.2501	0.2095
HILL	0.1629	0.2755	0.1389	0.2160
MiPOD	0.1412	0.1580	0.1812	0.1441

4.2.4.10 Detection performance of the proposed M-CNet for cover-source mismatch

Table 4.14: *Detection error P_E for proposed model cover-source mismatch for WOW 0.4 bpp*

Trained on	Tested on	P_E
Imagenet	BOSSbase	0.3936
BOSSbase ∪ BOWS2	Imagenet	0.4725

In order to assess the performance of the M-CNet for the scenario when training images are coming from one distribution and test images from another with the same embedding and payload (WOW 0.4 bpp). We performed the following set of experiments: (1) Trained on the training set explained in Section 4.2.2.1 (here, we referred to it as BOSSbase ∪ BOWS2) and tested on $5,000$ randomly

selected images from *Imagenet*[1] [7]. (2) Trained on 10,000 real-world images from the *Imagenet* dataset and tested on the BOSSbase test set. Since the *Imagenet* dataset consists of a wide variety of images, when the M-CNet is trained on *Imagenet* and tested on BOSSbase performed better than when it is trained on BOSSbase ∪ BOWS2 and tested on Imagenet. The results of this experiment are tabulated in Table 4.14.

4.3 Chapter Summary

In this contributory chapter, two novel steganalysis methods have been presented. In the first module of this chapter, a densely connected convolution network for steganalysis is presented. The model captures complex dependencies that are more appropriate for steganalysis, and the learned features avoid the loss of stego signals. The proposed model has no fully connected layer, which adds the advantage that the model can be tested on any image size, unlike with fully connected layers where the image size used for training and testing must be the same. The proposed scheme's steganalytic performance is compared with SRM, SPAM with EC, and a recent method [19] against different steganographic algorithms on different embedding rates. The proposed model outperforms the existing methods with a considerable margin.

In the second module of this chapter, a steganalytic detector, M-CNet, is proposed to learn the feature for the steganographic embeddings with multiple context sizes. The proposed M-CNet model is equipped with learned kernels for preprocessing, which offers diverse noise residual by exposing noise components and suppressing image components. Further, the M-CNet employed the *Self-Attention* mechanism to focus on the image regions, which are likely to be more affected by steganographic embeddings. Through an ablation study, the design of the proposed model is justified in favour of its detection performance. The

[1]Imagenet images are first converted to grayscale using *rgb2gray* and then resized to 256 × 256 using *imresize* function of Matlab before the embedding.

experimental results show that the M-CNet outperformed the state-of-the-art detectors.

In this contributory chapter, we have learned that a deeper network with efficient design criteria improves steganalytic detection accuracy. However, a deeper network may not always yield better detection accuracy. Instead, a proper balance between width and the depth of a network may help improve detection accuracy. In our third contributory chapter, we will consider this issue for further improving the detection accuracy.

Chapter 5

Steganalysis using Deep Fractal Network: The Role of Depth and Width

In the recent steganalysis literature, it has been observed that a deeper network using skip connections generally helps to model the steganographic noise efficiently [4]. Very recently, it has been reported that models with wider networks can carry more significant features through their layers [16]. Interestingly, it has been observed in *Wider Residual Network* [65] that a deep network where depth directly depends on the chosen width, performs better than any only-wider or only-deeper networks. To show this, the authors of [65] have conducted various experiments by varying the depth and width of the residual network on CIFAR-10 and CIFAR-100 datasets [114]. It has been observed that for a deep network, if we increase the width, the test error starts decreasing up to a certain extent; after that, it starts increasing [65]. Therefore, exploiting this depth and width trade-off, an optimal depth and width combination can be found that can maximize the detection accuracy.

Motivated by these observations, the following contributions have been made in this chapter:

- A concept of a recent deep-learning-based model popularly known as FractalNet [66] has been extended to design the proposed network to model the

steganographic noise where embedded images are used as input.

- The proposed model has been designed in such a way that a balance between the width and depth of the network can be maintained to maximize the detection performance.

5.1 Proposed Method

This section presents the details of the proposed method. The proposed model is inspired by FractalNet [66]. For simplicity, we call the proposed model as SFNet (*Steganalysis with Fractal architecture*).

5.1.1 FractalNet.

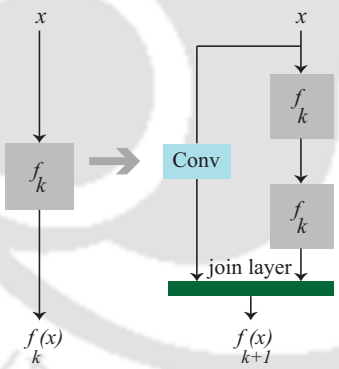


Figure 5.1: *Fractal expansion rule*

In recent years, extensive research has been done for image recognition tasks [64, 67, 70, 71]. Recently, FractalNet [66] has shown a competitive performance to the ResNet [64] for image recognition task. The FractalNet architecture is based on self-similarity and is generated by expanding the basic fractal/block by using the expansion rule shown in Figure 5.1. Formally, let k be the number of intertwined columns or width. The base case $f_1(x)$ contains a single layer of

convolution between input and output. i.e., $f_1(x) = conv(x)$. Successive fractals can be recursively defined using the following rule:

$f_{k+1}(x) = [f_k \circ f_k(x)] \oplus [conv(x)]$,

where $\circ$ represents the composition, and $\oplus$ denotes a join operation between two different blocks. The value k denotes the width of the network. The depth of the network, which is the longest path between the initial and final layer, can be defined as 2^{k-1}. This organization of FractalNet forces the network to vary the depth and width of the network proportionally. The *join layer* combines the features from two or more incoming paths. The features from the incoming connections can be joined using *sum*, *max-out*, *average*, or *concatenate*. In the latter case, the number of channels in the subsequent layers may increase. Further, FractalNet also uses drop-path regularization to force each input to a *join* layer to be individually significant. Further, the FractalNet is comprised of a number of fractal blocks connected using pooling layers. A detailed discussion on FractalNet can be found in [66].

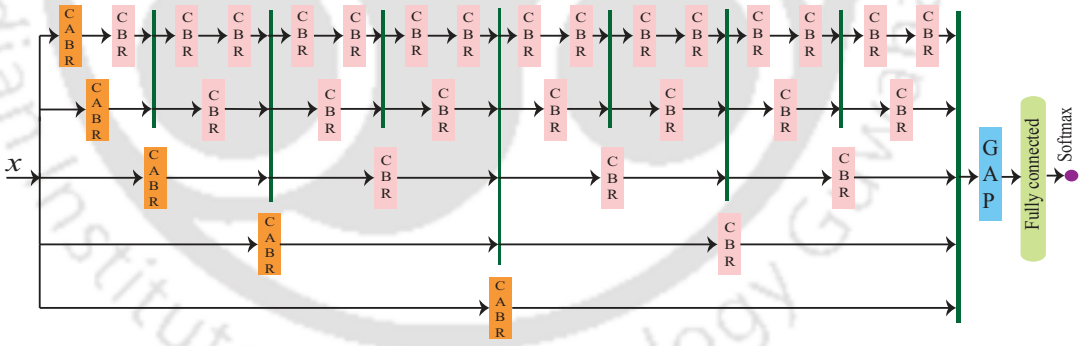


Figure 5.2: *An overview architecture of the proposed SFNet with no. of columns(n) = 5 and depth (d) = 16. However, the results mentioned in this chapter are with n = 6 and d = 32.*

5.1.2 SFNet Architecture

In this work, the concept of the FractalNet has been used. More specifically, the proposed network grows by using the expansion rule of the FractalNet architecture, where the balance between the depth and the width of the network

(i) C-A-B-R block *(ii) C-B-R block*

Figure 5.3: *C-A-B-R and C-B-R blocks of the proposed SFNet*

is maintained. In the proposed SFNet, two types of fundamental blocks have been used, namely - C-A-B-R and C-B-R. The detailed architecture of SFNet is shown in Figure 5.2, which consists of C-A-B-R and C-B-R blocks in a particular arrangement, followed by a fully connected classification module and join layers for connecting two fractals for expansion. The regimes of operations of C-A-B-R and C-B-R blocks are as follows.

C-A-B-R block is a sequence of a convolution layer with 16 filters followed by the ABS layer followed by *Batch Normalization* (BN) [58] and ReLU non-linearity at the end. The ABS layer after the convolutional layer allows the features with negative as well as positive values by discarding the sign of the generated feature map, which facilitates and improves statistical modeling of noise residual in the subsequent layers. These blocks are attached to the front-end of the network, which directly receives the input image of size 256×256 and outputs a feature map of the size $16 \times 256 \times 256$. An overview of the C-A-B-R block is shown in Figure 5.3 (i).

C-B-R block is a sequence of a convolution layer with 16 filters followed by BN [58] and ReLU non-linearity. The C-B-R block is shown in Figure 5.3 (ii). C-B-R block receives inputs either from a C-A-B-R block or from a previous C-B-R block, a feature map with dimension $16 \times 256 \times 256$, and outputs the feature with the same dimensionality. These blocks are the fundamental component of SFNet and are placed between the C-A-B-R blocks and fully connected layers.

In general, it is not possible to assert which task is being executed by which component of the deep CNN [4]. Therefore, it is difficult to describe the mapping function learned by the C-A-B-R and C-B-R blocks. However, the proposed

fractal-based architecture may have learned more significant features than any wider or deeper network for steganalysis by maintaining the balance between height and width, thereby achieving the competitive results with state-of-the-art.

Global Average Pooling (GAP) has been used for reducing the dimensionality of the feature space, which takes $16 \times 256 \times 256$ dimensional features and reduces it to $16 \times 1 \times 1$ dimensions. The output of the GAP is further fed to the fully connected layer, followed by softmax for binary classification.

Each fractal unit is expanded by using a *join* layer where an element-wise mean is computed between two incoming features. For example, let $\boldsymbol{P} = \{\boldsymbol{p_1}, \boldsymbol{p_2}, .., \boldsymbol{p_r}\}$ be the incoming features from one branch and $\boldsymbol{Q} = \{\boldsymbol{q_1}, \boldsymbol{q_2}, .., \boldsymbol{q_r}\}$ be the number of feature coming from another branch, the *join* layer computes the feature-wise arithmetic mean as $\boldsymbol{F} = \frac{1}{2}\sum_{i=1}^{r}(\boldsymbol{p_i} + \boldsymbol{q_i})$, where r is the number of features from each branch (number of the features from each branch is the same). In general, for n branches, each with r features, the output feature of the *join* layer is $\boldsymbol{F} = \frac{1}{n}\sum_{i=1}^{r}(\boldsymbol{p_i} + \boldsymbol{q_i})$. *Join* layers are shown using a vertical green line in the SFNet architecture, Figure 5.2.

Similar to the FractalNet, the width (k) and depth (d) of the proposed SFNet are related as follows

$$d = 2^{k-1} \tag{5.1}$$

The total number of blocks (*C-A-B-R* and *C-B-R*) can be calculated as: $N = \sum_{i=0}^{k} 2^i = 2^k - 1$. We have performed several experiments by varying the value of k from 2 to 7 and it has been observed that the SFNet performed best with $k = 6$, which means the depth d, and the total number of blocks N are 32 and 63, respectively.

5.2 Experimental Study

This section presents the experimental study, including - datasets, training and testing regimes, and the metric used to evaluate the proposed model.

5.2.1 Datasets.

The training and testing of the proposed SFNet are carried out on the union of BOSSBase 1.01 [85] and BOWS2 [86] dataset, each of which contains 10,000 grayscale images, each with size 512×512. The steganalytic performance of the SFNet is primarily compared with SRNet [4]. Therefore experimental setup has been kept similar to the SRNet. Each image is resized to 256×256 using Matlab *imresize* function. From BOSSBase, 4,000 images are randomly selected, and entire BOWS2 images have been used for training. From the remaining BOSSBase images, 1,000 are randomly chosen for validation, and the rest 5,000 are used for testing. The stego images are created using the steganography tools[1] with random keys. Therefore, the dataset comprised of $14,000 \times 2$ (cover and stego) used for training, $1,000 \times 2$ for validation, and 5000×2 for testing.

5.2.2 Training SFNet.

The proposed SFNet has been trained for spatial domain steganography schemes, namely- S-UNIWARD [15], WOW [5], HILL [18], and MiPOD [104] using Pytorch [99] framework on Nvidia V100 GPU (32 GB). The batch size of 20 cover / stego pairs (batch size =40) is used for training, validation, and testing. Each batch size is subject to data augmentation with random rotation by 90° and vertical/horizontal flip. The batch normalization parameter used as $\epsilon = 10^{-5}$, and momentum for running mean and running variance computation are set to 0.1. The kernel weights of each convolutional and fully connected layers are initialized with random normal distribution with $\mu = 0$ and $\sigma = 0.01$. Since batch normalization is used, biases of the kernels of convolutional layers are kept false [58]. The biases of the fully connected layers are initialized with zero. The training of

[1]Steganography algorithms can be downloaded from http://dde.binghamton.edu/download/stego_algorithms/

SFNet is carried out by minimizing the following loss function

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}(y_i.log\hat{y}_i + (1-y_i).log(1-\hat{y}_i)), \tag{5.2}$$

where y, $\hat{y}$, and N denote true label, predicted label, and the total number of training samples, respectively. Adamax [100] optimizer is used for optimization. The training is performed for 300 epochs (210k iterations). The learning rate is initialized to $lr = 10^{-3}$ and decayed every 25 epochs by a factor of 2. The model with the best validation accuracy is selected for the testing.

5.2.3 Comparison with state-of-the-art detectors.

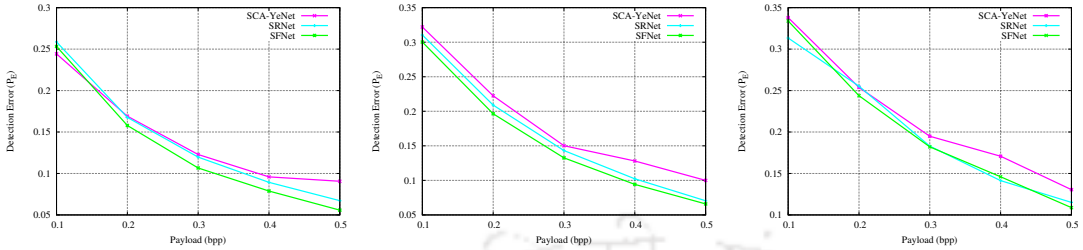
In order to evaluate the steganalytic performance of the proposed SFNet, it has been compared with the state-of-the-art detectors in the spatial domain, SR-Net [4] and YeNet [60]. Both the works are implemented with the experimental setup stated in the respective papers.

The results obtained using the proposed model, shown in Section 5.3, are given for a random 50-50 split of the BOSSBase dataset. This is because the in-detail experiments are infeasible, considering the resource and time complexity. However, to assess the performance across different BOSSBase splits, we trained the SFNet on five different 50-50 splits of BOSSBase (BOWS2 is kept fixed in the training set) for WOW at 0.5 bpp. The standard deviation of ≈ 0.00373 on P_E is found for these five splits.

5.3 Results

This section presents the results of the steganalytic experiments conducted for the proposed model.

The results of the steganalytic detection is reported for WOW, S-UNIWARD, and HILL for payloads: {0.1, 0.2, 0.3, 0.4, 0.5} bpp (bits-per-pixel). The steganalytic detection error P_E is given in Table 5.1, and for better comprehension, the



(a) WOW (b) S-UNIWARD (c) HILL

Figure 5.4: *Graphical plot of detection error probability on WOW, S-UNIWARD, and HILL on 0.1-0.5 bpp.*

Table 5.1: *Comparison of detection error P_E of the proposed scheme with the state-of-the-art steganalysis schemes. Best results are shown in* ***bold face****.*

Embedding	Detector	0.1	0.2	0.3	0.4	0.5
WOW	SCA-YeNet	0.2442	0.1691	0.1229	0.0959	0.0906
	SRNet	0.2587	0.1676	0.1197	0.0893	0.0672
	SFNet	**0.2532**	**0.1579**	**0.1066**	**0.0788**	**0.0558**
S-UNI	SCA-YeNet	0.3220	0.2224	0.1502	0.1281	0.1000
	SRNet	0.3104	0.2090	0.1432	0.1023	0.0705
	SFNet	**0.3002**	**0.1964**	**0.1326**	**0.0942**	**0.0659**
HILL	SCA-YeNet	0.3380	0.2538	0.1949	0.1708	0.1305
	SRNet	**0.3134**	0.2353	0.1830	**0.1414**	0.1151
	SFNet	0.3339	**0.2438**	**0.1821**	0.1460	**0.1088**

same results are graphically plotted in Figure 5.4. The proposed SFNet attains the improvement over YeNet [60] and SRNet [4] by $\sim 3\%$ and $\sim 1\%$, respectively, subject to the different embedding algorithms and payloads. A large improvement is obtained for the S-UNIWARD and WOW algorithms, whereas for HILL, we get performance comparable to SRNet. Further, as an alternative measure of evaluation, we have also given ROC curve in Figure 5.5 along with the AUC in Table 5.2.

5.3.1 Curriculum Learning.

One of the biggest challenges in training any steganalytic model is the time it takes to converge for the images with a low payload, such as 0.1 or 0.05 bpp. Sometimes the model failed to converge at all. In recent literature, this problem is usually

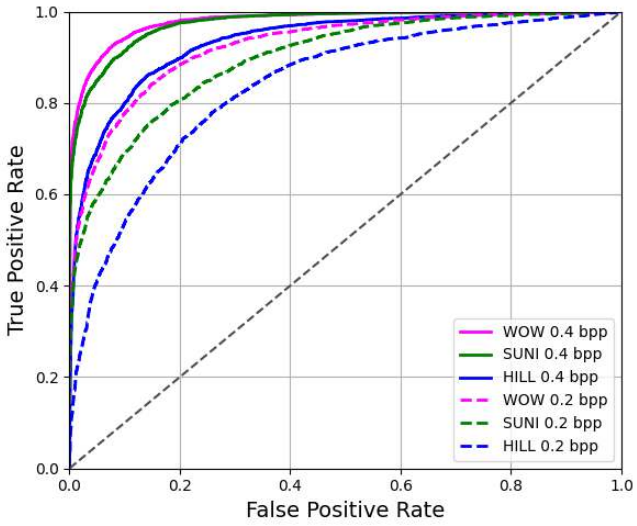


Figure 5.5: *ROC curves of SFNet for WOW, S-UNIWARD and HILL with payloads {0.2, 0.4} bpp.*

Table 5.2: *Area under ROC curve (AUC) for WOW, S-UNIWARD and HILL at payload ={0.2, 0.4} bpp for ROC curves plotted in Figure 5.5.*

Embedding	Payload (bpp)	
	0.2	0.4
WOW	0.9242	0.9787
S-UNI	0.8942	0.9835
HILL	0.8348	0.9333

handled by *Curriculum learning* [111], where the model is initially trained with easy examples (embedded with higher payload) and then gradually increased the difficulty level of the examples. Keeping these facts in mind, SFNet is initially trained for payload 0.4bpp and 0.5bpp for each steganographic algorithms, and then the learned weights with 0.5 bpp are transferred to train the model with lower payloads (0.1 - 0.3 bpp). The learning rate is kept very small (10^{-5}) during the finetuning for the images with smaller payloads. When finetuning, the model is trained up to 200 epochs, and the one with the best validation score is selected for testing. The detailed results are shown in Table 5.1 and Figure 5.4.

5.4 Ablation Study

This section presents the ablation study of the proposed model. Throughout this study, unless stated explicitly, we have used WOW embedding with payload 0.5 bpp to obtain corresponding stego images for all the experiments.

5.4.1 How does the SFNet architecture differs from the Fractal net?

The only similarity of the SFNet with the Fractal net [66] is the expansion rule and, more specifically, SFNet can be viewed as a block of Fractal net with no. of columns=6 modified with steganalysis point of view. However, the original Fractal net design consists of several such blocks (best, with no. of columns=4 and no. of fractal blocks=5) attached using max pooling. The Fractal net is designed for image classification applications. However, image classification, where the difference between the different classes of the image can be easily observed by the *Human Visual System* (HVS), is very different from the steganalysis, where one cannot perceive the difference between the cover and the corresponding stego images with HVS. To verify this effect, we trained the Fractal net (with no. of blocks=5, and no. of columns=4) with standard parameters reported in the paper [66] for 300 epochs using the dataset used for the experiment of SFNet. This configuration of the Fractal net failed to train with the dataset (training accuracy≈50% and validation accuracy ≈50%.). We found that this was because of the use of drop-path and dropout regularizations (too much regularization) used in the network. To overcome this situation, we trained the Fractal net without drop-path and dropout regularization. This modified Fractalnet overfitted the training data (training accuracy≈ 99% and validation accuracy≈ 50%) due to too complex architecture.

5.4.2 How does the choice of model architecture affect the performance?

An exhaustive set of experiments with different configurations of SFNet has been conducted in order to come up with the final model. The details of the experiments are as follows:

C-A-B-R vs. C-B-R block. To investigate the effect of applying ABS to the initial layers, we evaluated the model on WOW 0.5 bpp with and without ABS. The P_E without ABS was 0.0713, and with ABS, it was found to be 0.0596, which may be due to the ABS in initial layers offers better modeling of noise residuals [39]. We have also experimented with ABS throughout the network, which resulted in 0.0043↑[1] a rise in detection error, which means that adding ABS layer throughout the network, does not offer any better residual modeling and adds computational overhead. Therefore, the C-A-B-R block is kept only in the initial layers.

Number of filters in each layer. In contrast, the number of filters used in each convolution layer of the FractalNet [66] is 64. When SFNet is experimented with the 64 and 32 filters in each convolution layer, the steganalytic detection error probability achieved are 0.0932 (0.0336↑) and 0.0770 (0.0174↑) respectively. The best result is achieved for the number of filters in each convolution layer =16. This may be due to the fact that the proposed model is already wider, and by increasing the number of filters in each convolution layer, it may learn redundant features across the width. Therefore, all the values reported in Table 5.3 are using 16 filters in each convolution layer.

Width (k), Kernel size (s), and Activations. We started the process of selecting the model with very shallow architecture ($k = 2$) and varied the size of filters and activation functions (ReLU and TanH). We have used the filters of size

[1]↑ denote the increase (Bad) and ↓ decrease (Good) in P_E.

Table 5.3: *Variation of error detection probability with choice of architecture.*

Width	depth	s = 3		s = 5		s = 7	
$k =$	$d =$	ReLU	TanH	ReLU	TanH	ReLU	TanH
2	2	0.3401	0.4128	0.3774	0.3678	0.3597	0.4038
3	4	0.1823	0.2831	0.1812	0.2830	0.1997	0.3424
4	8	0.0821	0.1660	0.1254	0.1861	0.2090	0.3274
5	16	0.0791	0.1082	0.1025	0.1426	0.1336	0.2172
6	32	**0.0596**	0.1102	0.1024	0.1270	0.1942	0.2213
7	64	0.0698	0.1197	0.4093	0.4285	0.4650	0.4773

$s = \{3, 5, 7\}$, and the number of filters in each convolution layer was fixed to 16. Table 5.3 shows the result of steganalytic detection error by varying the k, s, and activation functions. Initially, increasing the width of the network from $k = 2$ to $k = 6$ resulted in performance gain but declined beyond $k = 6$ (for $k = 7$). The kernel size $s = 3$ is found to be the best choice for the network. However, when the kernel size for the convolution of C-A-B-R block except for the first one is varied from $s = 3$ to $s = 5$, a slight performance gain of $0.0038\downarrow$ is found ($P_E = 0.0558$), and this gain was consistent with other embedding algorithms as well. Among all the cases, SFNet performed better with the ReLU activation.

5.4.3 How does the SFNet behave when input is preprocessed using filters with fixed kernels?

The literature of steganalysis before the SRNet [4] heavily relied on the fixed high-pass filters such as KV filter, SRM filters [14], Gabor filters [62], etc., for preprocessing. To this end, we experimented using these filters to assess the behavior of SFNet when it is trained in the residual noise domain instead of the embedded image domain. The input images are preprocessed using these filters to get residual noise, which is fed to the model for training and evaluation. The preprocessing filters with smaller dimensions (less than 5×5) are padded with zeroes to match the dimension of 5×5 before preprocessing. The experimental results are shown in Table 5.4. It can be observed from the results, KV filter (a single 5×5 filter) offers less diverse noise residuals, which increases the P_E. When

preprocessing is done by using SRM or Gabor filters, the value of P_E relatively decreases. However, without preprocessing, the SFNet obtains the best P_E.

Table 5.4: *Steganalytic detection error when input images are preprocessed using hand-crafted filters. The best result is shown in* ***bold face****.*

Preprocessing with (filters)	No. of filters	Detection error probability
KV + SFNet	1	0.1038
SRM linear + SFNet	16	0.0698
SRM non-linear + SFNet	14	0.0673
SRM linear & non-linear + SFNet	30	0.0643
Gabor + SFNet	16	0.0632
SFNet	0	**0.0558**

5.4.4 How does SFNet perform for stego source mismatch?

In order to investigate the effects on the performance of the SFNet when the stego source is mismatched, a real-world situation, when the stego image embedding is not the same as the model trained on, we trained the model on one embedding algorithm and tested it on another. Table 5.5 shows the results when SFNet is trained on one steganography algorithm and tested on another. The results are shown for WOW, HILL, S-UNIWARD, and MiPOD for 0.4bpp. When SFNet is trained on an easily-detectable algorithm (WOW), it performs worse for the detection of the least-detectable embedding (MiPOD) and vice-versa.

5.4.5 How does SFNet perform for cover source mismatch?

Finally, the performance of the SFNet is evaluated for the situation of cover source mismatched, where the model is trained on different cover/stego pair datasets and tested on a different dataset. To this end, in one experiment, we trained the SFNet on the union of BOSSBase 1.01 [85] and BOWS2 [86] as original training stated in Section 5.2 and tested on the 2×5000 cover-stego pairs of

Table 5.5: *Detection error P_E for proposed model stego source mismatch for payload 0.4 bpp*

Train / Test	WOW	HILL	S-UNI	MiPOD
WOW	0.0788	0.2853	0.1369	0.2606
HILL	0.1484	0.1433	0.1891	0.1781
S-UNI	0.0996	0.2228	0.0942	0.1981
MiPOD	0.1426	0.1748	0.1655	0.1503

Table 5.6: *Detection error P_E for SFNet for cover source mismatch at 0.4 bpp.*

Train: ImageNet Steganography Algo.	**Test**: BOSSBase (P_E)
WOW	**0.3863**
MiPOD	**0.4473**
Train: BOSSBase Steganography Algo.	**Test**: ImageNet (P_E)
WOW	0.4569
MiPOD	0.4700

a real-world dataset, *ImageNet* [7]. In another experiment, we trained the SFNet on $2 \times 10,000$ cover-stego pairs of *ImageNet* and tested on 2×5000 images of the BOSSBase dataset. The *ImageNet* dataset has 1,000 classes of color images. From each class, 10 images with a dimension greater than or equal to 256×256 were randomly selected to form 10,000 images. These images are then resized to 256×256 and converted to grayscale using MATLAB functions *imresize* and *rgb2gray*. The results of these experiments are given in Table 5.6. It can be observed from results when SFNet is trained on the *ImageNet* and tested on the BOSSBase; it performs better steganalytic detection than vice-versa. It may be due to the diversity of the *ImageNet* dataset. Another observation can also be made that the performance of SFNet is consistent with other results for easily detectable WOW and hard to detect MiPOD.

Through the experimental and ablation studies, it has been shown that a balanced trade-off between the height and width of the SFNet performed better for

steganalytic detection. Furthermore, the proposed SFNet does not require any preprocessing using fixed filters such as KV or SRM filters.

5.5 Chapter Summary

In this third contributory chapter, a novel steganalysis scheme, SFNet, has been proposed, which is inspired by the concept of Fractal network. The proposed SFNet is an end-to-end network that does not involve any preprocessing filters to expose stego noise; and instead, it directly trains on the embedded images. The fractal architecture of SFNet allows the network to grow with a balance between depth and width, thereby achieving more accurate detection performance. SFNet models the stego features using C-A-B-R and C-B-R blocks without any residual shortcuts. The SFNet is tested with easily detectable as well as hard to detect steganographic algorithms and compared with the state-of-the-art steganalyzers. The experimental results reveal that the SFNet outperformed the state-of-the-art steganalysis schemes.

As of now, in the three contributory chapters, we have shown how a deeper model helps to get improved detection performance for spatial images. In the next and final contributory chapter, we will show how a GAN-based model can be used for hiding a complete image within another image without much visual degradation.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Chapter 6

StegGAN: Hiding Image within Image - An Application of Steganalysis

In the previous three contributory chapters, we have presented different deep-learning-based spatial image steganalysis techniques, which mostly outperform state-of-the-art literature. In this chapter, we will try to use a deep learning model for efficient data hiding. The research motivation for proposing efficient data hiding is mainly two-fold. Primarily, it helps different data hiding applications like digital watermarking, access control, covert communication, etc. In addition, it can challenge existing steganalytic detectors, which in turn helps to design more efficient steganalysis techniques.

From literature presented in Section 1.4.1.3, it has been observed that most of the *Generative Adversarial Network* (GAN) based data hiding methods emphasize perceptual imperceptibility and accurate extraction. Statistical undetectability has not been paid much importance. In addition to that, the extraction of the hidden message is not always precise due to the following reasons:

1. Intuitively, in steganography, both the embedder and extractor models play a two-player game in such a way that the extractor's response can be used to finetune the embedder effectively. However, vice-versa may not be much useful in training of the extractor model. This hypothesis has not been taken into consideration properly in most of the existing works [47, 48].

This not only reduces "Extraction" accuracy but, as an effect of in-accurate extraction, embedding performance is also reduced. Thus a majority of the existing solutions suffer from visual artifacts in the extracted hidden images.

2. Most of the methods mentioned above have taken only *Mean Squared Error* (MSE) into consideration as a loss function for estimating the stego images. However, in recent image restoration problems, it has been observed that the restored images suffer from visual artifacts if only MSE is used [115].

With the above line of thought, this work makes the following contributions.

1. We propose a two-players framework conceptually similar to a GAN, called "*Embedder - Extractor*," where both of them have an underlying architecture based on a cGAN framework.

2. The incorporated adversarial training in both networks helps to (1) generate stego image similar to cover (in case of proposed embedder) with respect to visual imperceptibility as well as statistical undetectability such that a recent steganalyzer XuNet [39] can be fooled, (2) retrieve hidden image indistinguishable to the original message (in case of the proposed extractor).

3. In addition to adversarial and MSE losses, the perceptual loss [116] has been used to train the proposed extractor to retrieve the hidden image with a better *Human Visual System* (HVS) quality.

4. A weighted loss from the proposed extractor has also been used to train the proposed embedder.

5. Finally, a thresholded version of the input cover image is given as a control map along with the cover image and a secret message.

6.1 Proposed Method

In this work, a Conditional Generative Adversarial Networks (cGAN) [37, 117] based model is proposed. For simplicity of reference, we call the proposed model

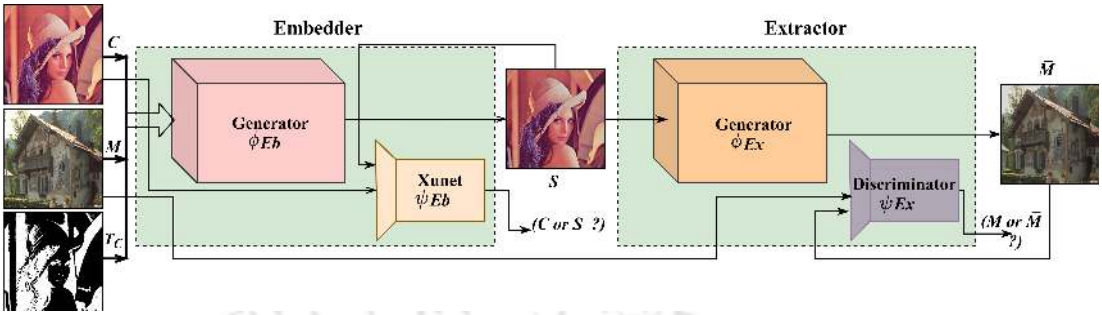


Figure 6.1: *An overview of the proposed framework for hiding an image within an image.*

StegGAN. The proposed model is comprised of two main modules : (a) An *embedder*, which hides the given secret image into the given cover image. (b) An *extractor*, which extracts the hidden secret message from the generated stego image by the embedder. Both the embedder and the extractor have an underlying architecture inspired by the cGAN, as shown in Figure 6.1. The embedder and the extractor networks consist of two sub-networks, namely a *Generator* and a *Discriminator*. Throughout this chapter, we denote ϕ_{Eb} and ψ_{Eb} as generator and discriminator of the embedder, respectively, and ϕ_{Ex} and ψ_{Ex} as generator and discriminator of the extractor, respectively. Given a cover image $\mathbf{C}$, a secret image $\mathbf{M}$, and the thresholded cover image $\mathbf{T_C}$, the embedder aims to hide the $\mathbf{M}$ into the $\mathbf{C}$ by using $\mathbf{T_C}$, such that the ϕ_{Eb} and ψ_{Eb} play a 2-player mini-max game until a Nash equilibrium [118] is achieved based on the following equation:

$$\min_{\phi_{Eb}} \max_{\psi_{Eb}} \mathcal{L}_{GAN}^{Eb}, \tag{6.1}$$

where $\mathcal{L}_{GAN}^{Eb} = \mathbb{E}_{\mathbf{C}\sim p_{cover}}[log(1-\psi_{Eb}(\phi_{Eb}(\mathbf{C},\mathbf{M},\mathbf{T_C})))] + \mathbb{E}_{\mathbf{M}\sim p_{secret}}[log(\psi_{Eb}(\mathbf{C}))]$. The proposed extractor aims to extract the $\mathbf{M}$ from the stego image $\mathbf{S}$ such that the ϕ_{Ex} and ψ_{Ex} play the similar two-player mini-max game based on the following equation:

$$\min_{\phi_{Ex}} \max_{\psi_{Ex}} \mathcal{L}_{GAN}^{Ex}, \tag{6.2}$$

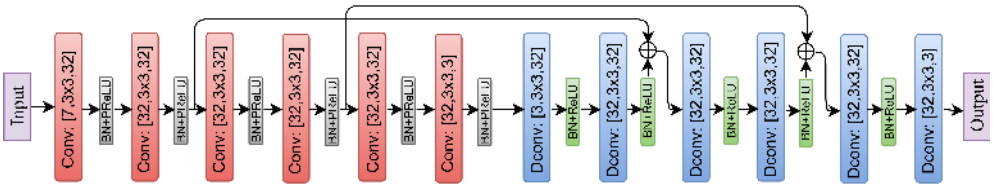


Figure 6.2: *The architecture of the encoder-decoder network used in the ϕ_{Eb} and ϕ_{Ex}. Each layer is labeled with $[a, b \times b, c]$, where a, b, and c represent no. of input channels, kernel-size, and no. of output channels, respectively.*

where $\mathcal{L}_{GAN}^{Ex} = \mathbb{E}_{\mathbf{S}\sim p_{stego}}[log(1 - \psi_{Ex}(\phi_{Ex}(S)))] + \mathbb{E}_{\mathbf{M}\sim p_{secret}}[log(\psi_{Ex}(\mathbf{M}))]$.

The regimes of operation of the ϕ_{Eb}, ψ_{Eb}, ϕ_{Ex}, and ψ_{Ex}, along with the given input is described as follows.

6.1.1 Network ϕ_{Eb}

Hiding an image into another image where both the images are assumed to be different in every context is a challenging task. It can be related to the image reconstruction and restoration problem where the reconstructed image is not only comprised of another hidden image in it but also maintains its originality. In recent times, deep models such as the encoder-decoder framework [41] have been useful in solving image restoration tasks. The generator sub-network ϕ_{Eb} of the embedder takes $\mathbf{C}, \mathbf{M}$, and $\mathbf{T_C}$ as input and predicts the stego image $\mathbf{S}$. The thresholded image $\mathbf{T_C}$ is computed as the *Otsu-thresholding* [119][1], with binary inverse for each cover image which gives the binary segmented image where the smooth regions are assigned value 0 and textured regions are assigned value 1. The ϕ_{Eb} sub-network consists of an encoder-decoder framework shown in Figure 6.2, where the encoder consists of 6 convolution layers. Each filter in the encoder part of the network has a spatial dimension of 3×3 with a stride and padding of 1. BN has been used for faster convergence of the network after every layer. Each convolution layer in the encoder part consists of 32 filters except the last layer, which has three filters. The decoder part consists of 6 transpose convolution

[1]Otsu-thresholding can be found at: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html

layers, where each layer consists of 32 filters except the final layer, which consists of 3 filters. The transpose convolution is also known as the "Deconvolution" operation, which is one of the most popular methods for upscaling the image in deep learning. Each transpose convolution layer in the decoder part is followed by BN and consists of kernels of size 3×3 with a spatial stride of 1. The layers in the encoder part include *Parametric ReLU* (PReLU) [120] as an activation function, whereas ReLU has been used in the decoder part.

6.1.2 Network ψ_{Eb}

The goal of the discriminator network is to maximize the probability of accurately classifying the samples into real (cover) or fake (stego) that inspires the embedder to generate a stego image more similar to the cover. The discriminator module, as shown in Figure 6.1, is inspired by the XuNet [39], which has been proposed to classify images into stego and cover.

6.1.3 Network ϕ_{Ex}

For simplicity, we have considered the architecture of the ϕ_{Ex} model similar to the ϕ_{Eb}, unlike in [45], [20], described earlier. The extractor model takes a stego image $\mathbf{S}$ as input and estimates the hidden secret image $\bar{\mathbf{M}}$. In general, a steganographic system uses a secret key that is used for embedding, which is shared with the intended receiver for extraction. In the proposed method, extractor network weights can be treated as a secret key, and it is assumed that weights have been shared offline with the receiver to extract the hidden message from the stego image [45].

6.1.4 Network ψ_{Ex}

The goal of the discriminator of the extractor network is to maximize the probability of classifying the samples into real ($\mathbf{M}$) or fake ($\bar{\mathbf{M}}$) extracted messages, thereby influence the extractor to estimate the message from the stego image

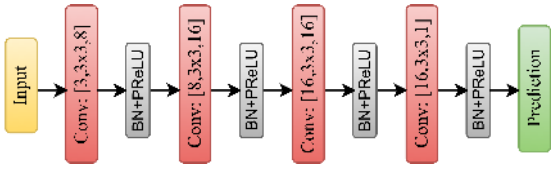


Figure 6.3: *The architecture of the ψ_{Ex}. Each layer is labeled with $[a, b \times b, c]$, where a, b, and c represent no. of input channels, kernel-size, and no. of output channels, respectively.*

more accurately. The discriminator model of the extractor network, as shown in Figure 6.3, consists of four convolution layers l_1, l_2, l_3, and l_4, with 8, 16, 16, and 1 kernel, respectively. The BN and PReLU activation follow each convolution layer. The network returns the mean sigmoid value of the output of the last convolution layer.

6.1.5 Cost Functions

The cost functions for the proposed models can be defined as follows:

6.1.5.1 Cost function of ϕ_{Eb}

The ϕ_{Eb} takes a cover and secret images in addition to a threshold map as input and estimates a stego image. The generated stego image should be identical to the cover image and also contains the secret message, which can be extracted in its original form later. The *Mean Squared Error* (MSE) is used to generate a stego image identical to the cover image, which can be defined as follows:

$$\mathcal{L}_2^{\phi_{Eb}} = ||\mathbf{S} - \mathbf{C}||_2^2 \tag{6.3}$$

The adversarial loss for N set of images from the discriminator ψ_{Eb} (XuNet) to guide the ϕ_{Eb} can be defined as

$$L_A^{\phi_{Eb}} = -\frac{1}{N}\sum_{i=1}^{N} log\ \psi_{Eb}(\phi_{Eb}(\mathbf{C}^i, \mathbf{M}^i, \mathbf{T}_\mathbf{C}^i)) \tag{6.4}$$

However, a model may learn to hide the secret image in the cover image better if it can learn from the extractor. The goal of the proposed model is that the extractor model can retrieve the hidden secret message from the stego image accurately despite that one can not differentiate between the stego and cover image both perceptually and statistically. Based on this argument, the ϕ_{Eb} takes the feedback from the extractor model, too, to compute its loss for optimization. The MSE between the extracted and actual secret message can be used to guide the ϕ_{Eb} and can be defined as

$$\mathcal{L}_E^{\phi_{Eb}} = ||\bar{\mathbf{M}} - \mathbf{M}||_2^2 \tag{6.5}$$

Therefore, the total loss for ϕ_{Eb} can be written as

$$\mathcal{L}_{\phi_{Eb}} = \mathcal{L}_2^{\phi_{Eb}} + \alpha.\mathcal{L}_A^{\phi_{Eb}} + \beta.\mathcal{L}_E^{\phi_{Eb}} \tag{6.6}$$

Consequently, the final training objective of the embedder model becomes:

$$\min_{\phi_{Eb}} \max_{\psi_{Eb}} \mathcal{L}_{GAN}^{Eb} + \mathcal{L}_2^{\phi_{Eb}} + \beta.\mathcal{L}_E^{\phi_{Eb}} \tag{6.7}$$

6.1.5.2 Cost function of ϕ_{Ex}

The extractor retrieves the hidden secret image from the stego image generated by the ϕ_{Eb}, and to achieve this, MSE is defined as:

$$\mathcal{L}_2^{\phi_{Ex}} = ||\bar{\mathbf{M}} - \mathbf{M}||_2^2 \tag{6.8}$$

However, MSE, in general, results in splotchy and blurred artifacts in the resultant images [121]. Therefore, to retain the high-frequency components in the reconstructed images, the perceptual loss[1] [116] is incorporated based on a pre-trained VGG16 [67] model ($\mathcal{V}$) at layer "relu2_2" which can be written as follows:

$$\mathcal{L}_{\mathcal{V}}^{\phi_{Ex}} = ||\mathcal{V}(\bar{\mathbf{M}}) - \mathcal{V}(\mathbf{M})||_2^2 \tag{6.9}$$

[1] https://github.com/pytorch/examples/blob/master/fast_neural_style/neural_style/neural_style.py.

Given a set of N estimated secret messages, the cross-entropy loss from the discriminator to regulate the ϕ_{Ex} is defined as:

$$\mathcal{L}_A^{\phi_{Ex}} = -\frac{1}{N}\sum_{i=1}^{N} log\ \psi_{Ex}(\phi_{Ex}(\mathbf{S}^i)) \tag{6.10}$$

Therefore, the total loss for the ϕ_{Ex} can be written as:

$$\mathcal{L}_{\phi_{Ex}} = \mathcal{L}_2^{\phi_{Ex}} + \kappa.\mathcal{L}_{\mathcal{V}}^{\phi_{Ex}} + \omega.\mathcal{L}_A^{\phi_{Ex}} \tag{6.11}$$

Consequently, the final training objective of the extractor becomes:

$$\underset{\phi_{Ex}}{min}\ \underset{\psi_{Ex}}{max}\ \mathcal{L}_{GAN}^{Ex} + \mathcal{L}_2^{\phi_{Ex}} + \kappa.\mathcal{L}_{\mathcal{V}}^{\phi_{Ex}} \tag{6.12}$$

The training algorithm for the proposed model is given as follows:

Algorithm 1 Training algorithm : Main

procedure MAIN($\mathbf{C}, \mathbf{M}$) ▷ Runs the ϕ_{Eb}, ϕ_{Ex} for embedding $\mathbf{M}$ in $\mathbf{C}$ and extracting $\bar{\mathbf{M}}$ from $\mathbf{S}$
    Initialize ϕ_{Eb}, ϕ_{Ex}, ψ_{Eb} and ψ_{Ex}.
    Get training data.
    Initialize α, β, κ and ω
    Initialize $\mathcal{V}$ ▷ Pretrained VGG for perceptual loss
    while No. of iterations **do**
        while Training data **do**
            $\mathbf{S} \leftarrow \phi_{Eb}(\mathbf{C}, \mathbf{M}, \mathbf{T_C})$
            $\bar{\mathbf{M}} \leftarrow \phi_{Ex}(\mathbf{S})$
            $\mathcal{O}(\phi_{Eb}(\bar{\mathbf{M}}, \mathbf{S}))$ ▷ Optimize the ϕ_{Eb} model
            $\mathcal{O}(\phi_{Ex}(\mathbf{S}, \bar{\mathbf{M}}))$ ▷ Optimize ϕ_{Ex} model
        end while
    end while
    return None
end procedure

6.2 Experiments

This section presents the details of the dataset, network parameters used to train the proposed framework, followed by the quantitative and the qualitative results.

Algorithm 2 Optimize algorithm : ϕ_{Eb}

procedure $\mathcal{O}(\phi_{Eb})(\bar{\mathbf{M}}, \mathbf{S})$ ▷ Update weights of ϕ_{Eb} i.e, $\theta_{\phi_{Eb}}$
  $\mathcal{L}_2^{\phi_{Eb}} \leftarrow ||\mathbf{S} - \mathbf{C}||_2^2$ ▷ Compute MSE loss
  $\mathcal{G}_2 \leftarrow \nabla \mathcal{L}_2^{\phi_{Eb}}$ ▷ Compute gradients w.r.t $\mathcal{L}_2^{\phi_{Eb}}$
  $\mathcal{L}_A^{\phi_{Eb}} = -\frac{1}{N} \sum_{i=1}^{N} log\ \psi_{Eb}(\phi_{Eb}(\mathbf{C}^i, \mathbf{M}^i, \mathbf{T}_\mathbf{C}^i))$ ▷ Compute Adv. loss
  $\mathcal{G}_A \leftarrow \nabla \alpha . \mathcal{L}_A^{\phi_{Eb}}$ ▷ Compute gradients w.r.t $\mathcal{L}_A^{\phi_{Eb}}$
  $\mathcal{L}_E^{\phi_{Eb}} = ||\bar{\mathbf{M}} - \mathbf{M}||_2^2$ ▷ Compute extractor feedback loss
  $\mathcal{G}_E \leftarrow \nabla \beta . \mathcal{L}_E^{\phi_{Eb}}$ ▷ Compute gradients w.r.t $\mathcal{L}_E^{\phi_{Eb}}$
  $\theta_{\phi_{Eb}} \leftarrow \theta_{\phi_{Eb}} - \{\mathcal{G}_2 + \mathcal{G}_A + \mathcal{G}_E\}$ ▷ Gradient descent
  Update weights of ψ_{Eb} i.e., $\theta_{\psi_{Eb}}$
  $\mathcal{L}_{Real} \leftarrow log(\psi_{Eb}(\mathbf{C}))$
  $\mathcal{G}_R \leftarrow \nabla \mathcal{L}_{Real}$
  $\mathcal{L}_{Fake} \leftarrow log(1 - \psi_{Eb}(\phi_{Eb}(\mathbf{C}, \mathbf{M}, \mathbf{T}_\mathbf{C})))$
  $\mathcal{G}_F \leftarrow \nabla \mathcal{L}_{Fake}$
  $\theta_{\psi_{Eb}} \leftarrow \theta_{\psi_{Eb}} + \mathcal{G}_R + \mathcal{G}_F$ ▷ Gradient ascent
  return None
end procedure

Algorithm 3 Optimize algorithm : ϕ_{Ex}

procedure $\mathcal{O}(\phi_{Ex})(\mathbf{S}, \bar{\mathbf{M}})$ ▷ Update weights of ϕ_{Ex} i.e, $\theta_{\phi_{Ex}}$
  $\mathcal{L}_2^{\phi_{Ex}} = ||\bar{\mathbf{M}} - \mathbf{M}||_2^2$ ▷ Compute MSE loss
  $\mathcal{G}_2 \leftarrow \nabla \mathcal{L}_2^{\phi_{Ex}}$ ▷ Compute gradients w.r.t $\mathcal{L}_2^{\phi_{Ex}}$
  $\mathcal{L}_\mathcal{V}^{\phi_{Ex}} = ||\mathcal{V}(\bar{\mathbf{M}}) - \mathcal{V}(\mathcal{M})||_2^2$ ▷ Compute perceptual loss
  $\mathcal{G}_\mathcal{V} \leftarrow \nabla \kappa . \mathcal{L}_\mathcal{V}^{\phi_{Ex}}$ ▷ Compute gradients w.r.t $\mathcal{L}_\mathcal{V}^{\phi_{Ex}}$
  $\mathcal{L}_A^{\phi_{Ex}} = -\frac{1}{N} \sum_{i=1}^{N} log\ \psi_{Ex}(\phi_{Ex}(\mathbf{S}^i))$ ▷ Compute adv. loss
  $\mathcal{G}_A \leftarrow \nabla \omega . \mathcal{L}_A^{\phi_{Ex}}$ ▷ Compute gradients w.r.t $\mathcal{L}_A^{\phi_{Ex}}$
  $\theta_{\phi_{Ex}} \leftarrow \theta_{\phi_{Ex}} - \{\mathcal{G}_2 + \mathcal{G}_\mathcal{V} + \mathcal{G}_A\}$ ▷ Gradient descent
  Update weights of ψ_{Ex} i.e., $\theta_{\psi_{Ex}}$
  $\mathcal{L}_{Real} \leftarrow log(\psi_{Ex}(\mathbf{M}))$
  $\mathcal{G}_R \leftarrow \nabla \mathcal{L}_{Real}$
  $\mathcal{L}_{Fake} \leftarrow log(1 - \psi_{Ex}(\phi_{Ex}(\mathbf{S})))$
  $\mathcal{G}_F \leftarrow \nabla \mathcal{L}_{Fake}$
  $\theta_{\psi_{Ex}} \leftarrow \theta_{\psi_{Ex}} + \mathcal{G}_R + \mathcal{G}_F$ ▷ Gradient ascent
  return None
end procedure

6.2.1 Training Details

The proposed framework is trained on randomly selected 0.1M images from the ImageNet [7] training dataset, out of which 50K is randomly chosen as cover images, and the rest were taken as secret images. Both cover and secret images have been resized to a spatial size of 256×256 during training.

The proposed model is trained for 191 epochs on NVIDIA Tesla V100 GPU (32 GB) using PyTorch [99]. The learning rate is experimentally reduced from 10^{-1} to 10^{-9}, the batch size of 16, and the Adam [100] optimization algorithm is used. The weights of the costs are experimentally set as follows: $\alpha = 0.2$, $\beta = 0.3$, $\kappa = 0.1$ and $\omega = 0.0004$. A variety of evaluation metrics, such as *Structural Similarity Index* (SSIM) [122], *Peak Signal to Noise Ratio* (PSNR), *Visual Information Fidelity* (VIF) [84], *Multi-scale Structural Similarity Index* (MS-SSIM) [1], and *Universal-Image-Quality Index* (UQI) [83] have been used to assess the proposed scheme. The proposed model is tested on a variety of datasets, namely- Imagenet [7], Microsoft COCO [8], DIV2K [9], KODAK [10], SIPI [11], and finally, with *Lena* image. For testing on Imagenet, a test set consists of randomly selected 1000 pairs of (cover, secret) images each of size 512×512 from the ImageNet [7], which have not been included in the training set is used.

6.2.2 Results

The quantitative results for the proposed model on the selected *Imagenet* test set are shown in Table 6.1. Though it is challenging to hide and retrieve an image from another image, the proposed embedder and extractor models have achieved a PSNR and SSIM of {42.24 dB, 0.9905}, and {37.17 dB, 0.9508}, respectively. In addition to traditional evaluation metrics PSNR and SSIM, the StegGAN has also achieved notable results in terms of VIF $\sim 0.87, 0.66$, MS-SSIM $\sim 0.99, 0.95$ and UQI $\sim 0.99, 0.93$ on the pairs (Cover, Stego) and (Secret, Extracted) respectively, as shown in Table 6.1. These findings prove the efficiency of the StegGAN in

Table 6.1: *Quantitative evaluation of the StegGAN on the test set selected from the Imagenet [7] test set. The best and the second-best results are shown in* ***Bold*** *and Blue fonts, respectively.*

Approach	Pairs	Statistics	SSIM	PSNR	VIF	MS-SSIM	UQI
w/o $\mathbf{T_C}$	C, S	μ	0.8096	18.7992	0.5131	0.8706	0.8014
		σ	0.0937	2.9920	0.0891	0.0493	0.1117
	M, $\bar{\text{M}}$	μ	0.7679	26.1422	0.4449	0.8658	0.7518
		σ	0.1406	2.1253	0.0959	0.0450	0.1924
w/o $\mathcal{L}_{\mathcal{V}}^{\phi_{Ex}}$	C, S	μ	0.9883	35.9531	0.8965	0.9906	0.9940
		σ	0.0057	2.6197	0.0380	0.0038	0.0075
	M, $\bar{\text{M}}$	μ	0.9437	36.2182	**0.6706**	0.9529	0.9147
		σ	0.0296	1.5264	0.0603	0.0123	0.1404
w/o $\mathcal{L}_{E}^{\phi_{Eb}}$	C, S	μ	**0.9911**	40.8910	**0.9301**	**0.9942**	0.9937
		σ	0.0131	1.7549	0.0244	0.0036	0.0177
	M, $\bar{\text{M}}$	μ	0.4105	10.5273	0.0054	0.5881	0.4568
		σ	0.1762	2.3492	0.0021	0.1008	0.2104
StegGAN	C, S	μ	0.9905	**42.2446**	0.8741	0.9902	**0.9985**
		σ	0.0033	0.9018	0.0393	0.0036	0.0027
	M, $\bar{\text{M}}$	μ	**0.9508**	**37.1736**	0.6618	**0.9542**	**0.9276**
		σ	0.0156	1.1595	0.0604	0.0102	0.1286

hiding an image within an image.

To justify the applicability of the proposed model, a few samples of the cover and secret images, corresponding stego, and extracted secret images from the Imagenet [7] test set are presented in Figure 6.4. The first through fourth column represents a cover, corresponding stego image, a secret, and corresponding extracted image, respectively. The fifth and sixth columns depict the residual computed at scale-×5 and intensities clipped at 255, between the cover and corresponding stego, and between the secret and corresponding extracted images. The rows show a variety of images hidden and extracted by using the proposed StegGAN. It can be observed from Figure 6.4 that the StegGAN achieves visually notable performance for a variety of images.

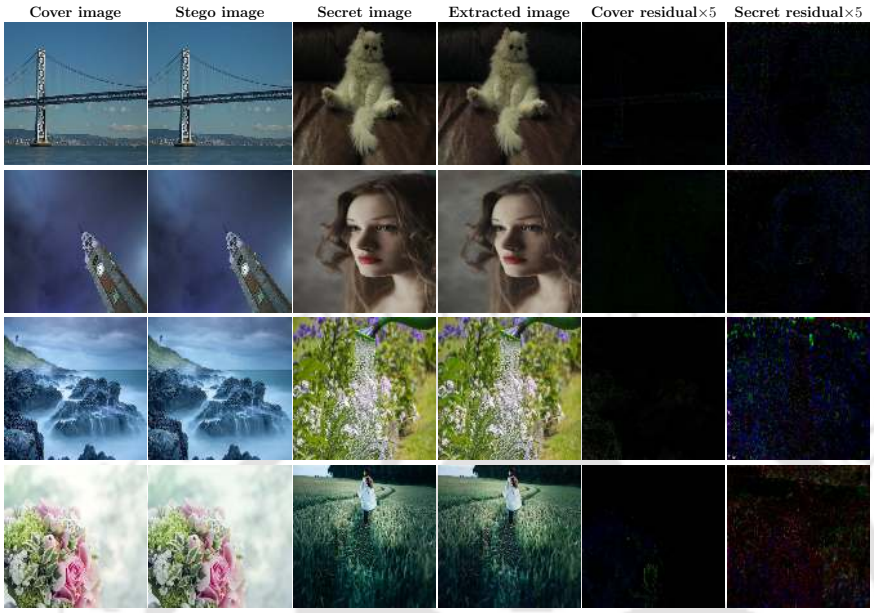


Figure 6.4: *Sample images from Imagenet [7]: cover images, stego images (Embedder output), secret images, extracted image, residual between the cover and stego ×5, and residual between secret and extracted ×5.*

Table 6.2: *Quantitative comparison of the StegGAN with [20] and [21] in terms of SSIM and PSNR for the cover and generated stego image.*

Metric	StegGAN	NIPS-17 [20]		T-PAMI-19 [21]	
		Reported	Implemented by us	Reported	Implemented by us
PSNR	42.24 dB	–	28.0764 dB	41.2 dB	28.5610 dB
SSIM	0.9905	–	0.9200	0.980	0.9216

6.2.3 Comparison with the existing schemes

For existing methods [20, 21], none of the following information is available: (i) Code/pre-trained model, (ii) Training/testing dataset, and (iii) Hyperparameters. However, to compare with these methods [20, 21], we have tried to implement the schemes with the best possible hyperparameters and datasets. With these hyperparameters, the trained model did not achieve the results as claimed in [20] and [21]. The results reported in [20] and [21], as well as the results of our implementation of [20] and [21], are given in Tables 6.2 and 6.3, along with the qualitative comparison as shown in Figure 6.5. Standard hyperparameters used:

Table 6.3: *Quantitative comparison of the StegGAN with [20] and [21] for the secret and extracted image.*

Metric	StegGAN	NIPS-17 [20]		T-PAMI [21]	
		Reported	Implemented by us	Reported	Implemented by us
PSNR	37.17 dB	–	24.7608 dB	37.6 dB	25.9725 dB
SSIM	0.9508	–	0.6773	0.9700	0.5955

Figure 6.5: *Sample images from Imagenet [7]: for qualitative comparison with existing schemes; $C = Cover$, $S = Secret$, $C' = Stego$ and $S' = Extracted$.*

learning rate- initialized to 10^{-1} and experimentally reduced till 10^{-9}, optimizer- Adam, batch size- 16, and no. of epochs- 200.

Our implementation could not achieve the result as claimed (41.2 dB between cover and stego and 37.6 between secret and extracted) in the paper. However, we were able to hide and extract the hidden secret image with some accuracy ((C, S) PSNR $\sim$ 28 dB) and ($(M, \bar{M})$ PSNR $\sim$ 25 dB).

6.2.4 Results on other Datasets

In this subsection, the test results of the StegGAN, which is already trained on Imagenet [7], have been presented on a variety of datasets, namely- Microsoft COCO [8], DIV2K [9], KODAK [10], and SIPI [11]. We have also presented the qualitative results when hiding in the *Lena* image. The StegGAN results are also compared with our implementations of [20] and [21] on these datasets.

6.2.4.1 Microsoft COCO and DIV2K

Object detection is often a more complex task than an image classification problem due to the contextual size of an object in an image. We have extended the

test of the generality of our proposed scheme on object detection tasks to verify if such types of images will be used to communicate then how our approach will handle such scenarios. For this, we have adopted the Microsoft COCO dataset, which is popular for object detection and segmentation. It has been observed that the proposed scheme achieves $\sim$ 0.98 in SSIM when compared to $\sim$ 0.85 and $\sim$ 0.80 by [20] and [21], respectively, on the **C**, **S** image pairs, as shown in Table 6.4. The qualitative comparison is also given in Figure 6.6. We have also tested our approach on the popular test-set, DIV2K, used for the *Single Image Super-Resolution* task, as shown in Table. 6.5. It can be observed from Table. 6.5 that the StegGAN obtains a remarkable improvement over the NIPS-17 [20] and T-PAMI [21] in terms of SSIM. The visual comparison is given in Figure 6.6.

Table 6.4: *Quantitative evaluation of the StegGAN on the test set selected from COCO [8] test set. The best and the second-best results are shown in* ***Bold*** *and Blue fonts, respectively.*

Approach	Image pairs	SSIM	PSNR	VIF	MS-SSIM	UQI
StegGAN	**C**, **S**	**0.9799**	**36.2609**	**0.8359**	**0.9980**	**0.9943**
	M, $\bar{\mathbf{M}}$	**0.9428**	**31.9235**	**0.6112**	**0.9774**	**0.9755**
NIPS-17 [20]	**C**, **S**	0.8481	25.5090	0.5645	0.9519	0.9723
	M, $\bar{\mathbf{M}}$	0.6581	23.4855	0.2136	0.7857	0.9311
T-PAMI [21]	**C**, **S**	0.8003	24.2799	0.5314	0.9536	0.9650
	M, $\bar{\mathbf{M}}$	0.5883	23.5594	0.1986	0.7526	0.9296

6.2.4.2 KODAK and SIPI

We have also tested our approach on two of the standard test sets in the domain of image processing, KODAK, and SIPI. The KODAK dataset has 25 uncompressed true-color images with dimensions of 768×512. These images are first resized to 512×512 and then arranged so that each image hides every other image ($25 \times 24 = 600$ image pair) for quantitative evaluation. From the SIPI dataset, 7 images (6 color and 1 grayscale) are selected from the "Miscellaneous" category, and 37 color images from the "Aerials" category (total 44 images) are chosen. These images are resized to 512×512 and arranged similarly to the KODAK

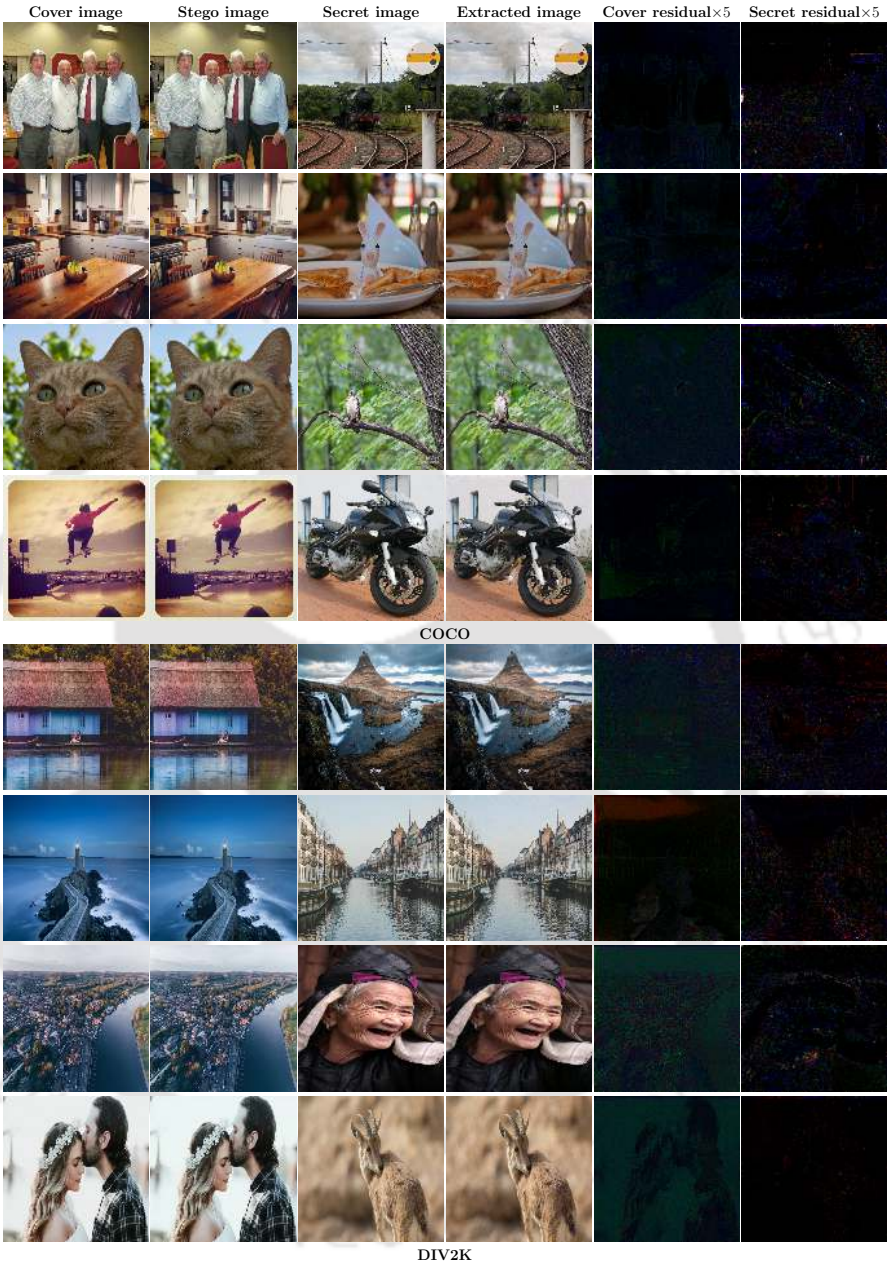


Figure 6.6: *Sample Images from COCO [8] and DIV2K [9] datasets: cover images, stego images, secret images, extracted image, residual between the cover and stego* $\times 5$ *and residual between secret and extracted* $\times 5$.

dataset for quantitative evaluation. The quantitative results are shown in the Tables. 6.6, and 6.7, respectively. It can be observed that the proposed scheme

Table 6.5: *Quantitative evaluation of the StegGAN on the test set selected from DIV2K [9] test set. The best and the second-best results are shown in* ***Bold*** *and Blue colors, respectively.*

Approach	Image pairs	SSIM	PSNR	VIF	MS-SSIM	UQI
StegGAN	**C, S**	**0.9744**	**34.3940**	**0.7898**	**0.9970**	**0.9935**
	M, $\bar{\mathbf{M}}$	**0.9410**	**31.8501**	**0.5909**	**0.9751**	**0.9903**
NIPS-17 [20]	**C, S**	0.7966	24.1730	0.5067	0.9483	0.9707
	M, $\bar{\mathbf{M}}$	0.5690	23.0901	0.1854	0.7572	0.9536
T-PAMI-19 [21]	**C, S**	0.8405	25.5222	0.5330	0.9580	0.9728
	M, $\bar{\mathbf{M}}$	0.6160	23.2747	0.1941	0.7845	0.9550

Table 6.6: *Quantitative evaluation of the StegGAN on the test set selected from KODAK [10] test set. The best and the second-best results are shown in* ***Bold*** *and Blue colors, respectively.*

Approach	Image pairs	SSIM	PSNR	VIF	MS-SSIM	UQI
StegGAN	**C, S**	**0.9788**	**30.9707**	**0.8064**	**0.9959**	**0.9942**
	M, $\bar{\mathbf{M}}$	**0.9363**	**29.4952**	**0.5553**	**0.9678**	**0.9920**
NIPS-17 [20]	**C, M**	0.7651	22.8523	0.4651	0.9383	0.9674
	M, $\bar{\mathbf{M}}$	0.5435	22.7666	0.1487	0.6877	0.9631
T-PAMI-19 [21]	**C, S**	0.8319	25.9525	0.5242	0.9377	0.9739
	M, $\bar{\mathbf{M}}$	0.6150	23.6198	0.1584	0.7330	0.9652

has not only outperformed the existing SoA methods for hiding an image within an image but also retains the visual features of the extracted images with ~ 0.99 in terms of UQI. The qualitative comparison is shown in Figure 6.7.

6.2.4.3 Hiding in *Lena* image

It is a widely known fact that the high-frequency details in an image are very difficult to reproduce during image reconstruction or restoration. For decades, *Lena* image, which comprises finer and high-frequency details with flat and symmetric face, has been used in image processing and computer vision. We also attemptd to use our approach to hide an image in the *Lena* image and observed that there is negligible loss of high-frequency details in the generated stego images and perceptually similar extracted images, as shown in Figure 6.8.

We have tested and compared the StegGAN on various test sets widely used

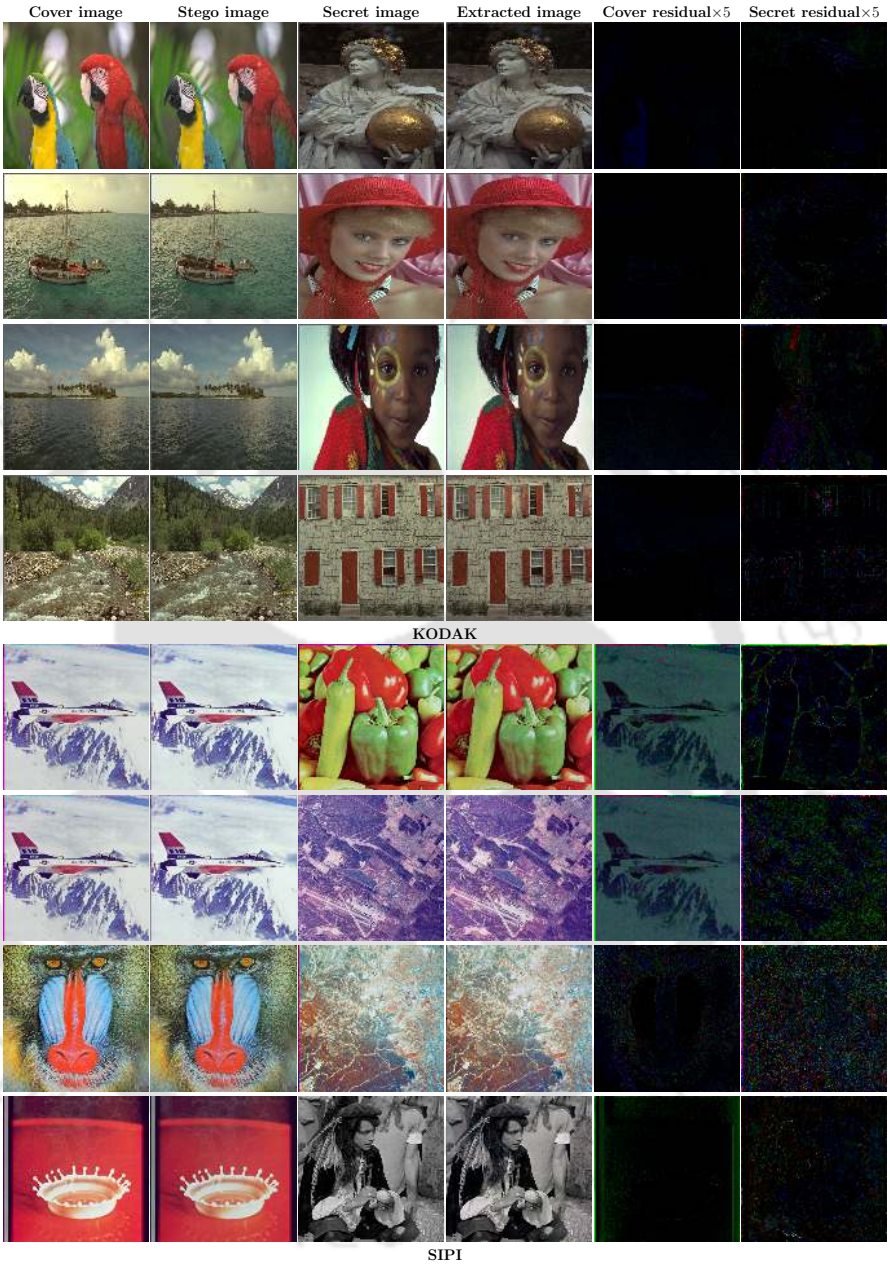


Figure 6.7: *Sample images from KODAK [10] and SIPI [11] datasets: cover images, stego images, secret images, extracted image, residual between cover and stego* $\times 5$ *and residual between secret and extracted* $\times 5$.

for different image restoration problems and observed that the StegGAN is more reliable for hiding an image within an image than the existing state-of-the-art

Table 6.7: *Quantitative evaluation of the StegGAN on the test set selected from SIPI [11] test set. The best and the second-best results are shown in* ***Bold*** *and Blue colors, respectively.*

Approach	Image pairs	SSIM	PSNR	VIF	MS-SSIM	UQI
StegGAN	**C, S**	**0.9788**	**30.9707**	**0.8064**	**0.9959**	**0.9942**
	M, $\bar{\mathbf{M}}$	**0.9363**	**29.4952**	**0.5553**	**0.9678**	**0.9920**
NIPS-17 [20]	**C, S**	0.7651	22.8523	0.4651	0.9383	0.9674
	M, $\bar{\mathbf{M}}$	0.5435	22.7666	0.1487	0.6877	0.9631
T-PAMI-19 [21]	**C, S**	0.8319	25.9525	0.5242	0.9377	0.9739
	M, $\bar{\mathbf{M}}$	0.6150	23.6198	0.1584	0.7330	0.9652

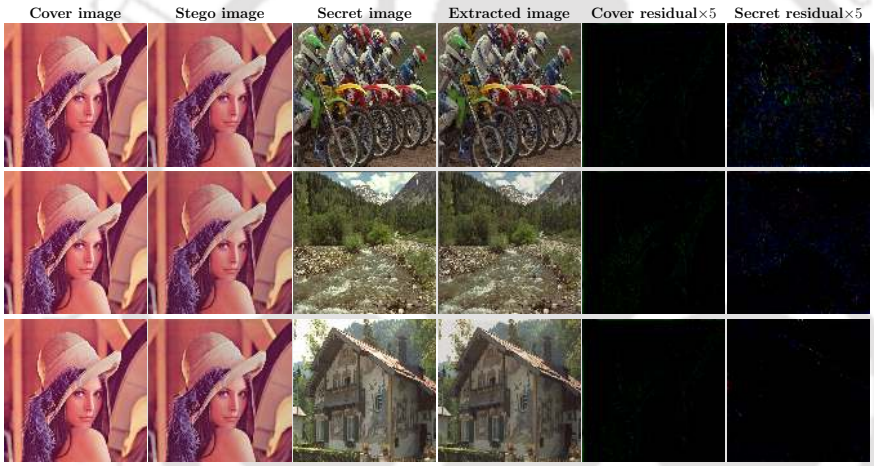


Figure 6.8: *Test results on a few KODAK images hidden in the Lena image.*

methods.

6.3 Ablation Study

In this section, an ablation study of the StegGAN has been presented. The randomly chosen subset of *Imagenet* [7], adopted as the test set for evaluating the proposed StegGAN in earlier sections, has also been utilized for different evaluations throughout this section.

Figure 6.9: *Comparison with baseline configurations.*

6.3.1 Comparison with the baseline configurations

The performance of the StegGAN has also been compared with the baseline configurations: (1) ***w/o*** $\mathbf{T_C}$: Similar to the proposed model except for input without threshold map $\mathbf{T_C}$, (2) ***w/o*** $\mathcal{L}_V^{\phi_{Ex}}$: Similar to proposed model except for the use of perceptual loss in the proposed extractor, (3) ***w/o*** $\mathcal{L}_E^{\phi_{Eb}}$: Similar to the proposed model except for the use of extractor loss in the training of embedder. The inclusion of the threshold map in the proposed embedder's input has improved the visual quality of both stego and extracted hidden images by $\sim$ +23.45 dB, $\sim$ +11.03 dB in PSNR, respectively, compared to its absence as in *w/o* $\mathbf{T_C}$. The thresholded input may have acted as a control signal, which might have regulated the intensity of the hidden image in such a way that it will be hidden in the high textured regions of the cover image while embedding. It has been observed that stego images generated in the case of *w/o* $\mathbf{T_C}$ model

suffer from visual artifacts (secret image outlines and color variation problem). The addition of the thresholded cover image as a control map has overcome this issue. Due to the fact that perceptual loss between features from the first several layers of the CNN helps in retaining the high-level features of the image, the proposed model outperforms baseline configuration *w/o* $\mathcal{L}_{\mathcal{V}}^{\phi_{Ex}}$ by $\sim$ +1 dB in PSNR. Intuitively, the feedback from the extractor to the embedder model has improved the visual quality of both stego and extracted hidden image by $\sim$ +1 dB, $\sim$ +26 dB in PSNR, respectively, as shown in Table 6.1.

The subjective comparison with the proposed baseline configurations has been shown in Figure 6.9. As mentioned above, it can also be observed from Figure 6.9 that the stego image in the case of *w/o* $\mathbf{T_C}$ suffers from the artifacts, whereas in the case of *w/o* $\mathcal{L}_{\mathcal{V}}^{\phi_{Ex}}$, the model has generated the noticeable results but secondary to the proposed scheme. In the absence of $\mathcal{L}_{E}^{\phi_{Eb}}$, even though the embedder generates the stego image almost similar to the cover image; the extractor model fails to retrieve the hidden secret image from the stego image.

6.3.2 Where is the secret image hidden within the image?

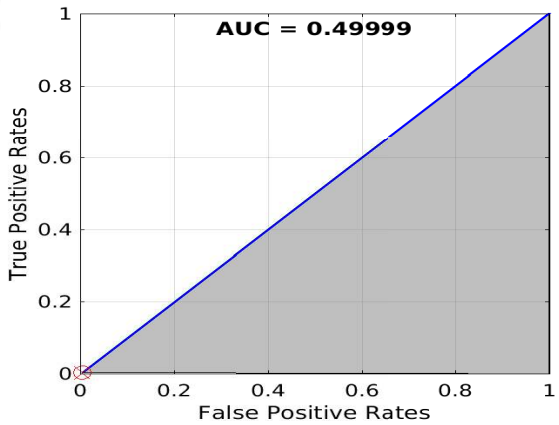


Figure 6.10: *ROC of StegExpose: ROC of steganalysis of embedded images using Stegoexpose [12]. From the curve, it can be observed that Stegoexpose unable to detect the stego images.*

When concealing an image in another image, one of the possibilities is, the secret image may get embedded in the LSB of the pixels of the cover image. The stego images generated from the proposed model are tested using a publicly available tool for LSB detection: *StegExpose* [12]. A 1000 pairs of cover and stego images were tested, the *StegExpose* with the default setting (threshold = 0.2). The *Receiver Operating Characteristics* (ROC) of the *StegExpose* shown in Figure 6.10 indicates that the *StegExpose* is not able to classify original images from the generated images beyond random guessing. This result (AUC$\approx$ 50) implies the model doesn't hide in the LSB of the cover. On the other hand, there are no visible artifacts such as ringing effects present in the stego image, which implies that it doesn't embed the image in MSB as well. Therefore, it can be inferred from the above observations that the proposed model may be using the intermediate bits of the pixels of the cover image to hide the secret image.

6.3.3 What amount of payload the *Embedder* introduce to the stego image?

In order to measure the amount of payload embedded by the Embedder when hiding a secret image in a cover image, we analyzed the 1000 cover images and corresponding embedded images. The average payload is found to be 1.7144 *bpp* ($\mu = 1.7144$ and $\sigma = 0.1255$). Though the proposed *Embedder* hides an entire secret image inside a cover image, it introduces the only payload of 1.7144 *bpp* to the cover image, which shows the efficacy of the proposed model over the existing scheme [20], whose embedding rate lies in [1.0 *bpp* − 4.0 *bpp*].

6.3.4 How much are the generated stego images robust against steganalytic detectors?

An obvious question for any steganography is how much secure the stego images are to the steganalytic detectors? Since the payload introduced by the proposed model is relatively high ($\sim$ 1.7 *bpp*) when hiding an entire image, we cannot claim that the steganalyzers cannot detect the stego images. To answer this

question, a popular steganalytic classifier, the *Color Rich Model* (CRMQ1) [51] with *Ensemble Classifier* (EC) [17], is trained and tested over 1000 generated pairs with 50-50 train and test split. The CRMQ1 [51] with EC was able to classify between a cover and a generated stego image with 89.22% accuracy.

6.3.5 How does a *Single image Layer Separation model* performs on the generated stego images?

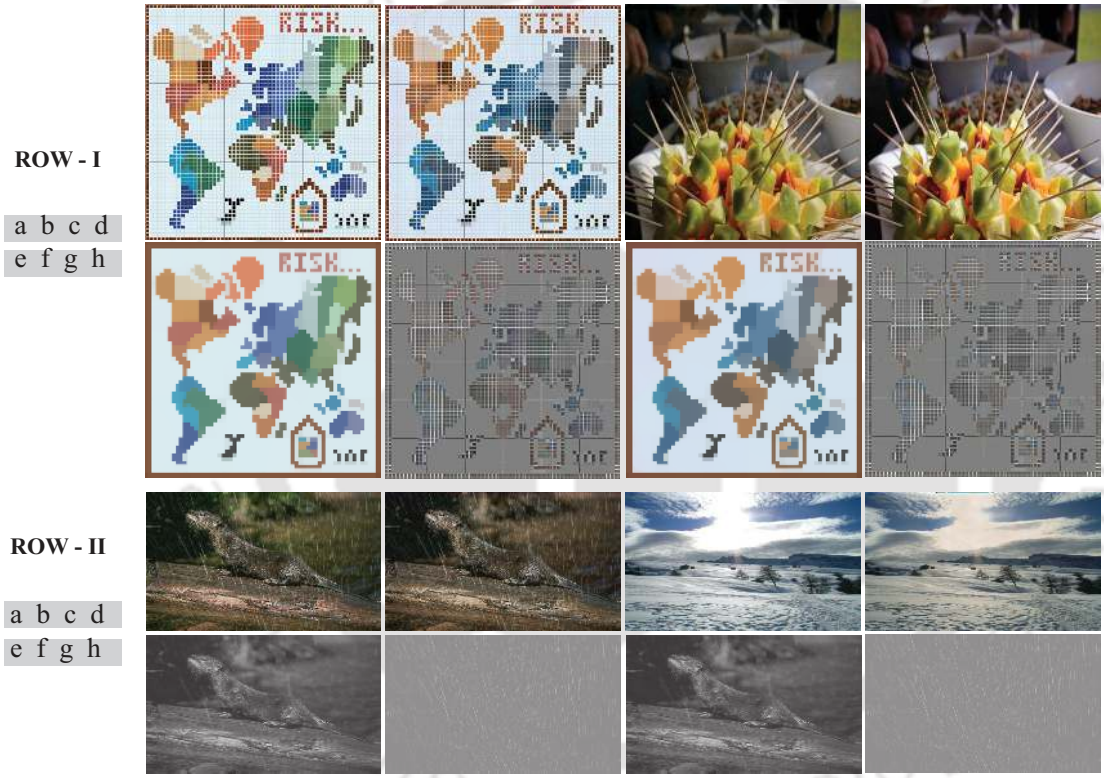


Figure 6.11: *Layer Separation: Row-I shows the layer separation of the Cartoon image* ***a.*** *original image* ***b.*** *embedded image,* ***c.*** *secret image,* ***d.*** *extracted secret,* ***e****, and* ***f*** *are layer separated image of* ***a****;* ***g*** *and* ***h*** *are layer separated images of* ***b****. Similarly, for the rainy image in Row-II.*

In literature, a layer separation problem is defined as a decomposition task where an image is decomposed into two layers. The problem of hiding an image in another image can be related to a layer super-imposition problem where an image is superimposed onto another image. However, such images can be well

separated by using layer separation techniques such as proposed in [123]. We use the proposed scheme in [123] and see whether the proposed model in this paper is performing super-imposition of the cover and secret images or steganography. To verify this, we take a cover image shown in Row-I as Figure 6.11 (a), a secret image is shown in Figure 6.11 (c) and generated the stego image by using the StegGAN as shown in Figure 6.11 (b). We then use the layer separation scheme proposed in [123] over the original cover image and generated the stego image one by one. The layers obtained by using [123] on the original cover image are shown in Figure 6.11 (e,f) in Row-I, whereas the same on the generated stego image is shown in Figure 6.11 (g,h). While layers of both stego and cover images look similar, note that no information of the hidden secret image can be inferred from either of the layers of stego image. This concludes experimentally that the StegGAN does not perform super-imposition of cover and secret images. Row-II of Figure 6.11 consists of another example where a secret image is hidden in a rainy image, and none of the layers of the stego image consists of any information about the hidden secret image.

6.3.6 Are embedded images sensitive to the *Wavelet Decomposition?*

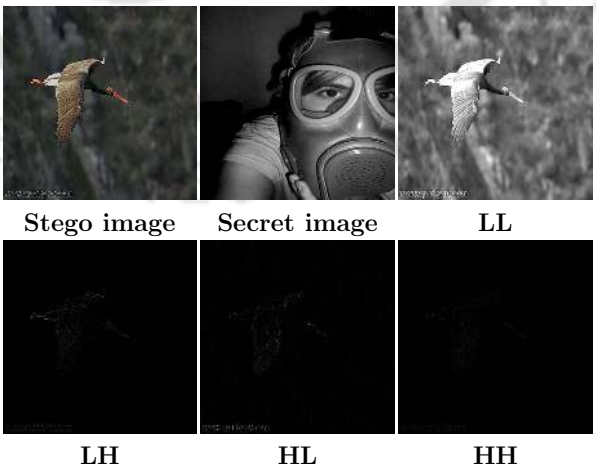


Figure 6.12: *Haar Wavelet decomposition of the stego image.*

In the previous subsection (Section 6.3.5), we have shown that the StegGAN does not perform super-imposition of cover and secret images using a *single image layer separation* method. To further support our claim, in this subsection, we have decomposed the stego image generated by the proposed model into the frequency domain sub-bands using *Haar Wavelet* transformation [124]. It can be observed from Figure 6.12 that none of the four sub-bands (LL, LH, HL, and HH) reveal any information about the hidden image. If the secret image is super-imposed on to the cover image, at least one of the sub-bands would have shown the edgy details of the secret image.

6.4 Chapter Summary

In this contributory chapter, a novel image hiding model, StegGAN, is proposed, which consists of two cGAN sub-networks. The first embedder sub-network is responsible for embedding an image within another image using adversarial training. The second extractor sub-network strives to extract the hidden image from the embedded image. A set of experiments has been carried out to justify the applicability of the proposed scheme over the state-of-the-art schemes. The experimental results reveal that the StegGAN outperformed the existing schemes. An ablation study is also given to justify the proposed architecture to achieve the presented results.

The next chapter concludes the thesis by briefly summarizing the work presented in the thesis and discussing the future research works.

Chapter 7

Conclusion and Future Works

7.1 Summary of the Contributions

In this dissertation, the main emphasis is on designing spatial image steganalysis based on deep learning. In the first three contributory chapters, it has been experimentally observed that convolutional neural network-based kernel learning for noise extraction performs better than fixed kernel-based methods. Multiple context-based image steganalysis model with self-attention yields better detection results. Moreover, maintaining a balance between the width and depth of a convolutional neural network achieves better detection accuracy. A GAN-based data hiding scheme is proposed in the final contributory chapter with a higher payload and reduced visual artifacts. The chapter-wise contributions are discussed below:

7.1.1 Breaking CMD Steganography and Kernel Learning for Steganalysis

In the first contributory chapter, several experiments are conducted to understand the relationship between the *Clustering Modification Directions* (CMD) embedding and the embedding location. *Selective-Signal-Removal* (SSR) method is proposed to counter the effect of the CMD algorithm. A comprehensive set of experiments shows that the proposed SSR method nullifies the effect of CMD embedding. Further, a deep learning method is proposed to learn a denoising

kernel that can replace the fixed high-pass filters used to preprocess the input images steganalytic classifier. It is empirically shown that the proposed method outperformed the existing steganalysis schemes.

7.1.2 Steganalysis: The Role of Hetrogeneous Context Size

In the second contributory chapter, the effect of heterogeneous context size is explored through two different methods. The first method investigates the impact of increasing the context size in each block, and then these blocks are densely connected to form a detector for steganalysis. This method also showed a way to detect the images with different resolutions. The second method proposed a multi-contextual design of CNN (MC-Net), where each block contains different context sizes. In order to amplify the signal-to-noise ratio of images, a set of thirty filters analogous to SRM filters is learned. The experimental results showed that the proposed method outperforms the state-of-the-art detectors.

7.1.3 Steganalysis using Deep Fractal Network: The Role of Depth and Width

The third contributory chapter explores the role of depth and width in designing steganalytic detectors. A steganalytic detector called SFNet is proposed that uses the concept of the deep fractal network to develop the proposed model. The depth and width of the model increased in a balanced way using a well-defined formula. This study showed that a deeper or a wider network performs well, but This study showed that a deeper or a wider network performs well, but a proper balance between width and depth makes the network more efficient with respect to steganalytic detection accuracy.

7.1.4 StegGAN: Hiding Image within Image - An Application of Steganalysis

In the final contributory chapter, a steganography method for hiding an image within another image is proposed as a deep learning application. The proposed model called StegGAN is based on the conditional GAN framework. StegGAN consists of an embedder, an extractor, two discriminators, one for embedder and another for the extractor. The discriminator of embedder is a well-known steganalyzer called XuNet. Unlike existing methods, StegGAN uses a thresholded version of the cover image to guide the model to embed in the high texture regions. Further, StegGAN minimizes various losses such as MSE, perceptual loss, extractor loss, and discriminator loss to perform the embedding. The proposed method is evaluated on a variety of datasets, and the results reveal that the StegGAN outperformed the existing methods quantitatively as well as qualitatively.

7.2 Future Works

The present study of this dissertation can be extended further in several directions as listed below:

- The denoising kernels proposed in chapters 3 and 4 can be learned by employing GAN for further improvements.
- All the steganalytic detectors proposed in this thesis are designed for the spatial domain. These works can be extended for JPEG domain steganalysis.
- StegGAN proposed in chapter 6 is presented to hide a single image within another image. This work can be further extended to hide and extract multiple images within a single image.

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

References

[1] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, vol. 2, Nov 2003, pp. 1398–1402 Vol.2. [Pg.xxiii], [Pg.30], [Pg.32], [Pg.116]

[2] Y. Qian, J. Dong, W. Wang, and T. Tan, "Deep learning for steganalysis via convolutional neural networks," in *Media Watermarking, Security, and Forensics 2015*, vol. 9409. International Society for Optics and Photonics, 2015, p. 94090J. [Pg.xxiv], [Pg.xxxi], [Pg.11], [Pg.14], [Pg.49], [Pg.56], [Pg.57], [Pg.69], [Pg.71], [Pg.83]

[3] J. Ye, J. Ni, and Y. Yi, "Deep learning hierarchical representations for image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2545–2557, 2017. [Pg.xxiv], [Pg.xxxi], [Pg.54], [Pg.56], [Pg.57]

[4] M. Boroumand, M. Chen, and J. Fridrich, "Deep residual network for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1181–1193, 2018. [Pg.xxiv], [Pg.xxxi], [Pg.3], [Pg.15], [Pg.56], [Pg.57], [Pg.59], [Pg.75], [Pg.77], [Pg.78], [Pg.79], [Pg.91], [Pg.94], [Pg.96], [Pg.97], [Pg.98], [Pg.102]

[5] V. Holub and J. Fridrich, "Designing steganographic distortion using directional filters," in *2012 IEEE International workshop on information forensics and security (WIFS)*. IEEE, 2012, pp. 234–239. [Pg.xxiv], [Pg.xxxi],

[Pg.xxxii], [Pg.9], [Pg.10], [Pg.35], [Pg.53], [Pg.56], [Pg.57], [Pg.58], [Pg.64], [Pg.67], [Pg.71], [Pg.75], [Pg.77], [Pg.96]

[6] T. Pevnỳ, T. Filler, and P. Bas, "Using high-dimensional image models to perform highly undetectable steganography," in *International Workshop on Information Hiding*. Springer, 2010, pp. 161–177. [Pg.xxiv], [Pg.xxxi], [Pg.xxxii], [Pg.9], [Pg.35], [Pg.53], [Pg.56], [Pg.57], [Pg.58], [Pg.64], [Pg.67]

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255. [Pg.xxvi], [Pg.xxxiii], [Pg.24], [Pg.33], [Pg.89], [Pg.104], [Pg.116], [Pg.117], [Pg.118], [Pg.119], [Pg.124]

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755. [Pg.xxvi], [Pg.xxxiii], [Pg.34], [Pg.116], [Pg.119], [Pg.120], [Pg.121]

[9] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017. [Pg.xxvi], [Pg.xxxiii], [Pg.116], [Pg.119], [Pg.121], [Pg.122]

[10] R. Franzen. Kodak lossless true color image suite. [Online]. Available: http://r0k.us/graphics/kodak/ [Pg.xxvi], [Pg.xxxiii], [Pg.116], [Pg.119], [Pg.122], [Pg.123]

[11] A. G. Weber, "The usc-sipi image database version 5," *USC-SIPI Report*, vol. 315, no. 1, 1997. [Pg.xxvi], [Pg.xxxiv], [Pg.116], [Pg.119], [Pg.123], [Pg.124]

[12] B. Boehm, "Stegexpose - A tool for detecting LSB steganography," *CoRR*, vol. abs/1410.6656, 2014. [Online]. Available: http://arxiv.org/abs/1410.6656 [Pg.xxvii], [Pg.126]

[13] T. Pevny, P. Bas, and J. Fridrich, "Steganalysis by subtractive pixel adjacency matrix," *IEEE Transactions on information Forensics and Security*, vol. 5, no. 2, pp. 215–224, 2010. [Pg.xxxi], [Pg.xxxii], [Pg.9], [Pg.12], [Pg.37], [Pg.56], [Pg.64], [Pg.67], [Pg.71]

[14] J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 868–882, 2012. [Pg.xxxi], [Pg.xxxii], [Pg.10], [Pg.12], [Pg.13], [Pg.37], [Pg.47], [Pg.56], [Pg.64], [Pg.67], [Pg.71], [Pg.82], [Pg.83], [Pg.102]

[15] V. Holub, J. Fridrich, and T. Denemark, "Universal distortion function for steganography in an arbitrary domain," *EURASIP Journal on Information Security*, vol. 2014, no. 1, p. 1, 2014. [Online]. Available: http://dx.doi.org/10.1186/1687-417X-2014-1 [Pg.xxxi], [Pg.xxxii], [Pg.9], [Pg.10], [Pg.35], [Pg.53], [Pg.56], [Pg.58], [Pg.64], [Pg.67], [Pg.71], [Pg.75], [Pg.77], [Pg.96]

[16] B. Li, W. Wei, A. Ferreira, and S. Tan, "Rest-net: Diverse activation modules and parallel subnets-based cnn for spatial image steganalysis," *IEEE Signal Processing Letters*, vol. 25, no. 5, pp. 650–654, 2018. [Pg.xxxii], [Pg.15], [Pg.56], [Pg.58], [Pg.59], [Pg.83], [Pg.91]

[17] J. Kodovsky, J. Fridrich, and V. Holub, "Ensemble classifiers for steganalysis of digital media," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 432–444, April 2012. [Pg.xxxii], [Pg.13], [Pg.66], [Pg.67], [Pg.127]

[18] B. Li, M. Wang, J. Huang, and X. Li, "A new cost function for spatial image steganography," in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct 2014, pp. 4206–4210. [Pg.xxxii], [Pg.9], [Pg.10], [Pg.64], [Pg.67], [Pg.71], [Pg.75], [Pg.77], [Pg.96]

[19] J. Tian and Y. Li, "Convolutional neural networks for steganalysis via transfer learning," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 33, no. 02, p. 1959006, 2019. [Pg.xxxii], [Pg.64], [Pg.67], [Pg.89]

[20] S. Baluja, "Hiding images in plain sight: Deep steganography," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 2069–2079. [Pg.xxxiii], [Pg.11], [Pg.111], [Pg.117], [Pg.118], [Pg.119], [Pg.120], [Pg.122], [Pg.124], [Pg.127]

[21] S. Baluja, "Hiding images within images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019. [Pg.xxxiii], [Pg.11], [Pg.117], [Pg.118], [Pg.119], [Pg.120], [Pg.122], [Pg.124]

[22] K. M. M. de Leeuw and J. Bergstra, *The history of information security: a comprehensive handbook*. Elsevier, 2007. [Pg.1]

[23] J. Mielikainen, "Lsb matching revisited," *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 285–287, 2006. [Pg.1], [Pg.8]

[24] G. J. Simmons, "The prisoners' problem and the subliminal channel," in *Advances in Cryptology*. Springer, 1984, pp. 51–67. [Pg.1]

[25] D. Giarimpampa, "Blind image steganalytic optimization by using machine learning," 2018. [Pg.3]

[26] R. Rodriguez-Colin, F.-U. Claudia, and G. d. J. Trinidad-Blas, "Data hiding scheme for medical images," in *17th International Conference on Electronics, Communications and Computers (CONIELECOMP'07)*. IEEE, 2007, pp. 32–32. [Pg.6]

[27] P. Wayner, *Disappearing cryptography: information hiding: steganography and watermarking*. Morgan Kaufmann, 2009. [Pg.6], [Pg.7]

[28] R. T. Mercuri, "The many colors of multimedia security," *Communications of the ACM*, vol. 47, no. 12, pp. 25–29, 2004. [Pg.7]

[29] H. Pang, K.-L. Tan, and X. Zhou, "Stegfs: A steganographic file system," in *Proceedings 19th International Conference on Data Engineering (Cat. No. 03CH37405)*. IEEE, 2003, pp. 657–667. [Pg.7]

[30] N. Provos and P. Honeyman, "Hide and seek: An introduction to steganography," *IEEE security & privacy*, vol. 1, no. 3, pp. 32–44, 2003. [Pg.8]

[31] X. Li, B. Yang, D. Cheng, and T. Zeng, "A generalization of lsb matching," *IEEE Signal Processing Letters*, vol. 16, no. 2, pp. 69–72, Feb 2009. [Pg.8]

[32] M. Khodaei and K. Faez, "New adaptive steganographic method using least-significant-bit substitution and pixel-value differencing," *IET Image processing*, vol. 6, no. 6, pp. 677–686, 2012. [Pg.8]

[33] Y.-K. Lee, G. Bell, S.-Y. Huang, R.-Z. Wang, and S.-J. Shyu, "An advanced least-significant-bit embedding scheme for steganographic encoding," in *Pacific-Rim Symposium on Image and Video Technology*. Springer, 2009, pp. 349–360. [Pg.8]

[34] V. Holub *et al.*, *Content Adaptive Steganography: Design and Detection*. Citeseer, 2014. [Pg.8]

[35] T. Filler, J. Judas, and J. Fridrich, "Minimizing additive distortion in steganography using syndrome-trellis codes," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 920–935, Sept 2011. [Pg.9]

[36] B. Li, M. Wang, X. Li, S. Tan, and J. Huang, "A strategy of clustering modification directions in spatial image steganography," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1905–1917, 2015. [Pg.10], [Pg.16], [Pg.17], [Pg.35], [Pg.43]

[37] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680. [Pg.10], [Pg.27], [Pg.108]

[38] W. Tang, S. Tan, B. Li, and J. Huang, "Automatic steganographic distortion learning using a generative adversarial network," *IEEE Signal Processing Letters*, vol. 24, no. 10, pp. 1547–1551, 2017. [Pg.10]

[39] G. Xu, H.-Z. Wu, and Y.-Q. Shi, "Structural design of convolutional neural networks for steganalysis," *IEEE Signal Processing Letters*, vol. 23,

no. 5, pp. 708–712, 2016. [Pg.10], [Pg.14], [Pg.19], [Pg.69], [Pg.71], [Pg.74], [Pg.83], [Pg.101], [Pg.108], [Pg.111]

[40] J. Yang, D. Ruan, J. Huang, X. Kang, and Y.-Q. Shi, "An embedding cost learning framework using gan," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 839–851, 2019. [Pg.11]

[41] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241. [Pg.11], [Pg.110]

[42] D. Volkhonskiy, I. Nazarov, B. Borisenko, and E. Burnaev, "Steganographic generative adversarial networks," *CoRR*, vol. abs/1703.05502, 2017. [Online]. Available: http://arxiv.org/abs/1703.05502 [Pg.11]

[43] S. Haichao, D. Jing, W. Wei, Q. Yinlong, and Z. Xiaoyu, "Ssgan: Secure steganography based on generative adversarial networks," in *Advances in Multimedia Information Processing – PCM 2017*. Cham: Springer International Publishing, 2018, pp. 534–544. [Pg.11]

[44] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017, pp. 214–223. [Pg.11]

[45] J. Hayes and G. Danezis, "Generating steganographic images via adversarial training," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 1954–1963. [Pg.11], [Pg.111]

[46] R. Zhang, S. Dong, and J. Liu, "Invisible steganography via generative adversarial networks," *Multimedia Tools and Applications*, Dec 2018. [Online]. Available: https://doi.org/10.1007/s11042-018-6951-z [Pg.11]

[47] J. Zhu, R. Kaplan, J. Johnson, and L. Fei-Fei, "Hidden: Hiding data with deep networks," in *The European Conference on Computer Vision (ECCV)*, September 2018. [Pg.11], [Pg.12], [Pg.107]

[48] K. A. Zhang, A. Cuesta-Infante, and K. Veeramachaneni, "Steganogan: Pushing the limits of image steganography," *arXiv preprint arXiv:1901.03892*, 2019. [Pg.12], [Pg.107]

[49] C. J. Geyer, "Practical markov chain monte carlo," *Statistical science*, pp. 473–483, 1992. [Pg.12]

[50] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Berlin, Heidelberg: Springer-Verlag, 1995. [Pg.12]

[51] M. Goljan, J. Fridrich, and R. Cogranne, "Rich model for steganalysis of color images," in *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, Dec 2014, pp. 185–190. [Pg.13], [Pg.127]

[52] J. Kodovskỳ and J. Fridrich, "Steganalysis of jpeg images using rich models," in *Media Watermarking, Security, and Forensics 2012*, vol. 8303. International Society for Optics and Photonics, 2012, p. 83030A. [Pg.13]

[53] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. [Pg.14]

[54] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. [Pg.14]

[55] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105. [Pg.14], [Pg.21], [Pg.24]

[56] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1299–1312, 2016. [Pg.14]

[57] S. Tan and B. Li, "Stacked convolutional auto-encoders for steganalysis of digital images," in *Signal and Information Processing Association Annual*

Summit and Conference (APSIPA), 2014 Asia-Pacific. IEEE, 2014, pp. 1–4. [Pg.14]

[58] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift. arxiv. org website," 2019. [Pg.14], [Pg.65], [Pg.94], [Pg.96]

[59] Y. Qian, J. Dong, W. Wang, and T. Tan, "Learning and transferring representations for image steganalysis using convolutional neural network," in *2016 IEEE international conference on image processing (ICIP)*. IEEE, 2016, pp. 2752–2756. [Pg.15], [Pg.78]

[60] J. Ye, J. Ni, and Y. Yi, "Deep learning hierarchical representations for image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2545–2557, 2017. [Pg.15], [Pg.69], [Pg.71], [Pg.75], [Pg.77], [Pg.78], [Pg.97], [Pg.98]

[61] L. Taylor and G. Nitschke, "Improving deep learning using generic data augmentation," *arXiv preprint arXiv:1708.06020*, 2017. [Pg.15]

[62] X. Song, F. Liu, C. Yang, X. Luo, and Y. Zhang, "Steganalysis of adaptive jpeg steganography using 2d gabor filters," in *Proceedings of the 3rd ACM workshop on information hiding and multimedia security*. ACM, 2015, pp. 15–23. [Pg.15], [Pg.82], [Pg.83], [Pg.102]

[63] M. Yedroudj, F. Comby, and M. Chaumont, "Yedroudj-net: An efficient cnn for spatial steganalysis," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 2092–2096. [Pg.15], [Pg.69]

[64] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. [Pg.15], [Pg.21], [Pg.25], [Pg.26], [Pg.92]

[65] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016. [Pg.16], [Pg.18], [Pg.91]

[66] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. [Pg.18], [Pg.21], [Pg.25], [Pg.91], [Pg.92], [Pg.93], [Pg.100], [Pg.101]

[67] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. [Pg.21], [Pg.24], [Pg.92], [Pg.113]

[68] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "Densenet: Implementing efficient convnet descriptor pyramids," *arXiv preprint arXiv:1404.1869*, 2014. [Pg.21], [Pg.65]

[69] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep fisher networks for large-scale image classification," in *Advances in Neural Information Processing Systems*, vol. 26, 2013, pp. 163–171. [Pg.21]

[70] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9. [Pg.21], [Pg.92]

[71] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826. [Pg.21], [Pg.92]

[72] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for natural language processing," *arXiv preprint arXiv:1606.01781*, vol. 2, 2016. [Pg.21]

[73] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 160–167. [Pg.21]

[74] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, pp. 649–657, 2015. [Pg.21]

[75] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," *Advances in neural information processing systems*, vol. 28, pp. 2224–2232, 2015. [Pg.21]

[76] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 4845–4849. [Pg.21]

[77] S.-W. Fu, Y. Tsao, X. Lu, and H. Kawai, "Raw waveform-based speech enhancement by fully convolutional networks," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2017, pp. 006–012. [Pg.21]

[78] V. Mitra and H. Franco, "Time-frequency convolutional networks for robust speech recognition," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2015, pp. 317–323. [Pg.21]

[79] T. Pevnỳ and A. D. Ker, "Towards dependable steganalysis," in *Media Watermarking, Security, and Forensics 2015*, vol. 9409. International Society for Optics and Photonics, 2015, p. 94090I. [Pg.29]

[80] W. You, H. Zhang, and X. Zhao, "A siamese cnn for image steganalysis," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 291–306, 2020. [Pg.29]

[81] R. Cogranne, Q. Giboulot, and P. Bas, "Alaska-2: Challenging academic research on steganalysis with realistic images," in *IEEE International Workshop on Information Forensics and Security*, 2020. [Pg.29]

[82] A. Horé and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th International Conference on Pattern Recognition*, 2010, pp. 2366–2369. [Pg.30]

[83] Z. Wang and A. C. Bovik, "A universal image quality index," *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81–84, March 2002. [Pg.30], [Pg.116]

[84] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, May 2004, pp. iii–709. [Pg.30], [Pg.32], [Pg.116]

[85] P. Bas, T. Filler, and T. Pevný, ""break our steganographic system": The ins and outs of organizing boss," in *Proceedings of the 13th International Conference on Information Hiding*, ser. IH'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 59–70. [Online]. Available: http://dl.acm.org/citation.cfm?id=2042445.2042452 [Pg.33], [Pg.43], [Pg.47], [Pg.52], [Pg.53], [Pg.64], [Pg.66], [Pg.75], [Pg.96], [Pg.103]

[86] T. F. Patrick Bas. (July, 2007) Bows-2. [Online]. Available: http://bows2.ec-lille.fr/ [Pg.33], [Pg.52], [Pg.53], [Pg.75], [Pg.96], [Pg.103]

[87] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017. [Pg.34]

[88] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C 23, no. 1, pp. 90–93, Jan 1974. [Pg.39]

[89] H.-J. Bae and S.-H. Jung, "Image retrieval using texture based on dct," in *Proceedings of ICICS, 1997 International Conference on Information, Communications and Signal Processing. Theme: Trends in Information Systems Engineering and Wireless Multimedia Communications (Cat.*, vol. 2, Sept 1997, pp. 1065–1068 vol.2. [Pg.39]

[90] Y. len Huang, "A fast method for textural analysis of dct-based image," *Journal of Information Science and Engineering*, pp. 181–194, 2005. [Pg.39]

[91] I. F. Iatan, "The fisher's linear discriminant," in *Combining Soft Computing and Statistical Methods in Data Analysis*, C. Borgelt, G. González-Rodríguez, W. Trutschnig, M. A. Lubiano, M. Á. Gil, P. Grzegorzewski, and O. Hryniewicz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 345–352. [Pg.45]

[92] S. Gujar and C. Veni Madhavan, "Measures for classification and detection in steganalysis," *ArXiv e-prints*, Jan. 2009. [Pg.47]

[93] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014. [Pg.52]

[94] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011. [Pg.53], [Pg.55]

[95] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005. [Pg.55]

[96] J. Fridrich. (2012) Digital data embedding laboratory. [Online]. Available: http://dde.binghamton.edu/download/ [Pg.56]

[97] M. Abadi, P. Barham, J. Chen *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283. [Pg.59]

[98] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814. [Pg.65]

[99] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration," *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, vol. 6, 2017. [Pg.66], [Pg.75], [Pg.96], [Pg.116]

[100] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. [Pg.66], [Pg.97], [Pg.116]

[101] B. Singh, P. K. Sharma, R. Saxena, A. Sur, and P. Mitra, "A new steganalysis method using densely connected convnets," in *International Conference on Pattern Recognition and Machine Intelligence*. Springer, 2019, pp. 277–285. [Pg.69], [Pg.71]

[102] B. Singh, M. Chhajed, A. Sur, and P. Mitra, "Steganalysis using learned denoising kernels," *Multimedia Tools and Applications*, pp. 1–15, 2020. [Pg.70], [Pg.71], [Pg.72]

[103] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7354–7363. [Pg.70], [Pg.73], [Pg.87]

[104] V. Sedighi, R. Cogranne, and J. Fridrich, "Content-adaptive steganography by minimizing statistical detectability," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 2, pp. 221–234, 2015. [Pg.71], [Pg.75], [Pg.96]

[105] B. Li, M. Wang, X. Li, S. Tan, and J. Huang, "A strategy of clustering modification directions in spatial image steganography," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1905–1917, 2015. [Pg.71]

[106] W. Tang, B. Li, W. Luo, and J. Huang, "Clustering steganographic modification directions for color components," *IEEE Signal Processing Letters*, vol. 23, no. 2, pp. 197–201, 2015. [Pg.71]

[107] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015. [Pg.74]

[108] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of*

the IEEE international conference on computer vision, 2015, pp. 1026–1034. [Pg.74], [Pg.81], [Pg.82], [Pg.87]

[109] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. [Pg.76]

[110] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256. [Pg.76], [Pg.82]

[111] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48. [Pg.78], [Pg.99]

[112] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in *Proceedings of ICML workshop on unsupervised and transfer learning*, 2012, pp. 17–36. [Pg.78]

[113] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3. [Pg.87]

[114] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 and cifar-100 datasets," *URl: https://www. cs. toronto. edu/kriz/cifar. html*, vol. 6, 2009. [Pg.91]

[115] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47–57, March 2017. [Pg.108]

[116] J. Johnson, A. Alahi, and F. Li, "Perceptual losses for real-time style transfer and super-resolution," *CoRR*, vol. abs/1603.08155, 2016. [Online]. Available: http://arxiv.org/abs/1603.08155 [Pg.108], [Pg.113]

[117] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014. [Pg.108]

[118] J. F. Nash, "Equilibrium points in n-person games," *Proceedings of the National Academy of Sciences*, vol. 36, no. 1, pp. 48–49, 1950. [Pg.109]

[119] X. Lang, F. Zhu, Y. Hao, and J. Ou, "Integral image based fast algorithm for two-dimensional otsu thresholding," in *2008 Congress on Image and Signal Processing*, vol. 3, May 2008, pp. 677–681. [Pg.110]

[120] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR*, vol. abs/1502.01852, 2015. [Online]. Available: http://arxiv.org/abs/1502.01852 [Pg.111]

[121] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for image restoration with neural networks," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47–57, March 2017. [Pg.113]

[122] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004. [Pg.116]

[123] S. Gu, D. Meng, W. Zuo, and L. Zhang, "Joint convolutional analysis and synthesis sparse representation for single image layer separation," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 1717–1725. [Pg.128], [Pg.129]

[124] G. Pajares and J. M. De La Cruz, "A wavelet-based image fusion tutorial," *Pattern recognition*, vol. 37, no. 9, pp. 1855–1872, 2004. [Pg.129]

भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
Indian Institute of Technology Guwahati

Appendix: A

Journal Publications:

- **Brijesh Singh**, Arijit Sur, and Pinaki Mitra. ***"Steganalysis of Digital Images using Deep Fractal Network."*** IEEE Transactions on Computational Social Systems. DOI: https://doi.org/10.1109/TCSS.2021.3052520

- **Brijesh Singh**, Mohit Chhajed, Arijit Sur, and Pinaki Mitra. ***"Steganalysis using Learned Denoising Kernels."*** Multimedia Tools and Applications, Springer. October 2020. DOI: https://doi.org/10.1007/s11042-020-09960-w

- Ritvik Rawat, **Brijesh Singh**, Arijit Sur, and Pinaki Mitra. ***"Steganalysis for clustering modification directions steganography."*** Multimedia Tools and Applications, Springer.January 2020. Volume 79, pp 1971–1986 (2020). DOI: https://doi.org/10.1007/s11042-019-08263-z

Conference Publications:

- **Brijesh Singh**, Prasen Kumar Sharma, Rupal Saxena, Arijit Sur, and Pinaki Mitra, ***"A New Steganalysis Method Using Densely Connected ConvNets."*** Pattern Recognition and Machine Intelligence. PReMI 2019. Lecture Notes in Computer Science, vol 11941. Springer. DOI: https://doi.org/10.1007/978-3-030-34869-4_31

Under Submission:

- **Brijesh Singh**, Prasen Kumar Sharama, Shashank Huddedar, Arijit Sur, and Pinaki Mitra, ***StegGAN: Hiding Image within Image using Conditional Generative Adversarial Networks.*** Multimedia Tools and Applications, Springer, February 2021. (Revision Submitted)
- **Brijesh Singh**, Arijit Sur, and Pinaki Mitra, ***Multi-Contextual Design of Convolutional Neural Network for Steganalysis.***, arXiv preprint arXiv:2106.10430 (2021). (Submitted to Multimedia Tools and Applications (Springer))