



KLE Technological
University
Creating Value
Leveraging Knowledge

BVB Campus, Vidyanagar, Hubballi – 580031, Karnataka, INDIA.

A Minor Project Report on

Medical Image Steganography

Submitted

in partial fulfillment of the requirements for the degree of

Bachelor of Engineering

in

Computer Science and Engineering

Submitted By

Deepak Shinde	01FE18BCS069
Debabrata Maity	01FE18BCS068
Arun Harikant	01FE18BCS050
Darshan Patgar	01FE18BCS066

Under the guidance of

Dr. Shrinivas Desai

Associate Professor

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

KLE Technological University, Hubballi

2020-2021



KLE Technological
University
Creating Value
Leveraging Knowledge

BVB Campus, Vidyanagar, Hubballi – 580031, Karnataka, INDIA.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2020-21

CERTIFICATE

This is to certify that project titled “Medical Image Steganography ” is a bonafied work carried out by the student team Mr. Debabrata Maity – 01FE18BCS068, Mr. Deepak Shinde - 01FE18BCS069, Mr. Arun Harikant - 01FE18BCS050, Mr. Darshan Patgar - 01FE18BCS066 for partial fulfillment of the completion of sixth semester B.E. in Computer Science and Engineering during the academic year 2020-21.

Guide Name
Dr. Shrinivas Desai

SoCSE Head
Dr. Meena S. M.

Viva-Voce

Name of the examiners

Signature with date

1 _____

2 _____

ABSTRACT

Steganography is the art of hiding an object within another object from the perspective of secure transmission. In this project, medical image steganography using deep neural network is presented. The medical image steganography is basically hiding a medical image into any cover image. The deep neural network is used for data preprocessing, hiding images as well as revealing the stego image which is carried out by preparation network, hiding network and revealing network respectively. The proposed method deep neural network based image steganography has proved to be better compared to the existing solutions as well as the conventional image steganography methods. The PSNR loss being recorded is 17.54% which is less than the loss that is recorded using conventional and latest techniques. Deep steganography improves upon existing methods for digital steganography as the classical methods are easy to decode and the amount of information that can be hidden is very limited.

Keywords : *Image Steganography, Medical image, Deep Neural Network, Covid-19*

ACKNOWLEDGEMENT

We would like to thank our faculty and management for their professional guidance towards the completion of the project work. We take this opportunity to thank Dr. Ashok Shettar, Vice-Chancellor, Dr. N.H Ayachit, Registrar, and Dr. P.G Tewari, Dean Academics, KLE Technological University, Hubballi, for their vision and support.

We also take this opportunity to thank Dr. Meena S. M, Professor and Head, SoCSE for having provided us direction and facilitated for enhancement of skills and academic growth.

We thank our guide Dr. Shrinivas Desai, Associate Professor, SoCSE for the constant guidance during interaction and reviews.

We extend our acknowledgement to the reviewers for critical suggestions and inputs. We also thank Project Co-ordinator Dr. Sujatha C. and faculty in-charges for their support during the course of completion.

We express gratitude to our beloved parents for constant encouragement and support.

Team Members :

01FE18BCS069

01FE18BCS068

01FE18BCS050

01FE18BCS066

CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENT	i
CONTENTS	iii
1 INTRODUCTION	1
1.1 Motivation	2
1.2 Literature Review / Survey	2
1.3 Problem Statement	4
1.4 Applications	4
1.5 Objectives and Scope of the project	5
1.5.1 Objectives	5
1.5.2 Scope of the project	5
2 REQUIREMENT ANALYSIS	6
2.1 Functional Requirements	6
2.2 Non Functional Requirements	6
2.3 Hardware Requirements	6
2.4 Software Requirements	7
3 SYSTEM DESIGN	8
3.1 System Design	8
3.2 Architectural Design	9
3.3 Design Principles	9
3.3.1 Design Principles 1	9
3.3.2 Design Principles 2	10
3.4 Basic Methodologies design principle	10
3.4.1 Least Significant Bit (LSB) Steganography	10
3.4.2 Discrete cosine transform	11
3.4.3 Discrete wavelet transform	11
3.4.4 Binary pattern complexity	13
3.5 Data structure used	14
3.6 Dataset Description	15

4	IMPLEMENTATION	16
4.1	Proposed Algorithm	16
4.2	Data Preprocessing	17
4.3	Generation of Deep Neural Network	17
4.3.1	Preparation Network	17
4.3.2	Hiding Network	18
4.3.3	Reveal Network	18
4.4	Model Integration	18
4.5	Loss Function	19
4.6	Loss Graph	19
4.7	Secret message error and cover message error	19
4.8	Histogram Generation	20
4.9	PSNR Calculation	20
4.10	SSIM Calculation	21
4.11	Correlation Calculation	21
4.12	Intersection Calculation	21
4.13	KL Divergence	22
5	RESULTS AND DISCUSSIONS	29
5.1	Basic Method Results	29
5.2	Comparative Analysis	30
6	CONCLUSION AND FUTURE SCOPE OF THE WORK	33
6.1	Conclusion	33
6.2	Future Scope	33
	REFERENCES	34

Chapter 1

INTRODUCTION

Modern steganography is the design and creation of hidden communications that create intangible changes in digital images to conceal confidential information without revealing secret information's existence to unintended users. In simple data, hiding is a technique that hides the confidential message in another medium such as image, audio, video format, etc. It is essential to hide the secret information so that the attacker or intruder cannot steal the message. We worked on Steganography which hides the secrete image from another image and also it is the process of embedding information within a medium in an imperceptible way. The art of hiding data is a precious task that only the sender and intended recipient suspect the presence of hidden data. Although the distortion functions are designed innovatively, steganography algorithms will be optimal an external will not be able to control the messages if he cannot end them. So, in this project, we will be dealing with the newly adapted security system known as deep steganography where the security level of data would be enhanced. Steganography is the art of hiding data within a medium without letting others know about the existence of data. Deep Steganography technique uses Deep Neural Network to achieve both embedding and decoding with the help of 3 different networks mainly preparation, hiding and reveal network. Deep steganography is a technique that involves the encoder and decoder architecture. The models are trained on various medical as well as the random dataset.

The challenge of good information hiding arises because encoding a message can change the appearance and underlying statistics of the stego image(carrier image). Total amount of alteration depends on two factors: i.e. , the amount of information needed to hide. Hiding relatively small number of bits is the most common form of information hiding tha exists . Potential distortion will be more on increasing the length of message . The other is , the amount of visible alteration depends on the stego image only. Hiding information in the noisy, high frequency, regions of an image yields less humanly detectable perturbations than hiding in the flat regions.

In this work, the deep neural network determines where to embed the hidden information, as well as how much to compress and represent it. The hidden image is basically distributed throughout the bits in all available pixels and across all the color channels. The decoder network that has been simultaneously trained with the encoder is used to reveal the secret

image from the stego image. All the networks are trained once and does not depend on the client and secret images. As they are trained as a pair, all networks work with each other. In the upcoming section, how the system of neural networks are simultaneously trained to hide and retrieve images will be described. Learning-based detectors can identify the presence of secret information by training. An analysis of a presentation of various methods and how the information is stored like conventional methods for retrieving the secret information are presented in this report.

1.1 Motivation

In the era of technology huge amount of data is manipulated over the internet. This data may consist of personal information of people, the Government's data, and the huge amount of social media records that need to be secured properly. In the era of confidential communication such as military and other Government sector areas, the data should be more secure against an attacker who steals the information. Nowadays the data is very huge and the medium needs to be more secure in communication. So many hackers found their way into economics for stealing information. As daily we are getting news of hackers hacking the data continuously for illegal use, it has become a need now to protect this data using some advanced technology where only desired user will be beneficiary of the content. So we need the Medical eld which needs a medium to communicate their information in a secure way.

1.2 Literature Review / Survey

In this paper [1], Author describes about a new algorithm for hiding data is implemented which can be used for encrypting grey images as well as medical color images. They introduced a new image dividing method based on various blocks of an image. Then, the image blocks disorganized using a rotation, random permutation, combination and zigzag pattern. Then, a chaotic logistic map is used to generate a key that will diffuse the disorganized image. Using the factors like time complexity and security analysis, the efficiency of this methodology in encrypting secret images has been evaluated.

Gaps identified :

- To increase the vulnerability of data by adding more number of map channels.

In this paper [2], Authors presents a stego medical image hiding technique using a nuclear spin generator system. Detailed experimental and theoretical analysis is provided to the proposed algorithm using peak signal-to-noise ratio, pixel difference histogram analysis, statistical package analysis and key space calculation. The result of the model show good performance compared to the latest available medical image steganographic technique.

Gaps identified :

- Improvement on embedding capacity and message stability will be the focus for future work.

This paper [3], the authors describe the edge detection in images. In this methodology the Canny Algorithm has been used. The proposed work has been done on the basis of swapped Hoffman tree coding (SHT) and encryption using the same. Further SHT coding is implemented for security analysis that shows the embedding of data and decoding process of the hidden or secret message.

Gaps identified :

- To handle the unbalancing in the learning of generator-discriminator can happen where the generator is performing efficiently but the discriminator is struggling.

In this paper [4], Author describes about a high gradual digital steganography method is implemented. It has ability to hide electronic patient records known as EPR into a medical cover image without altering any important region or part of data. This method uses embedded secret or hidden data in sharp regions of the image and edge detection. In the end they can hide EPR into non-interested region to hide the decision region or area which is necessary for diagnosis.

In this paper [5], Author presents a imperceptibility , robustness and high capacity in terms of various evaluation metrics like PSNR, MSE and execution time. They have achieved a high data hiding capacity by concealing data in the sub bands of high frequency. The high PSNR is obtained for Imperceptibility and robustness is achieved by embedding the random data in the cover image.

Gaps identified :

- Using transfer domain method instead of LSB method.

In this paper [6], The author presents imperceptibility, robustness and high capacity of steganography results in terms of PSNR, MSE and execution time. Concealment in the high frequency sub bands can lead to high data hiding capacity. They concluded that imperceptibility can be obtained through high PSNR whereas robustness can be through the random embedding of the data in an image.

Gaps identified :

- To approve the proposed approach by executing security attacks and performance analysis on messages.

1.3 Problem Statement

Many times, data transmission(image, audio, video etc.) from various sources are frequently carried out through the internet for different applications, such as personal photography, defense sector secret data circulation, confidential business archives, documentation of various data, medical images, patient details are embedded within image proving protection to information, and army imaging database. As these images contain extremely important and confidential data, they should be highly protected from piracy or leakage during transmissions. So, there is a need for a secure image transmission technique. With this in context, we define our problem statement as “To develop a model which can secure the medical images by Steganography using Deep Neural Network”.

1.4 Applications

- Confidential communication and secret data storing.
- Watermarking
- Government agencies store critical data like criminal record using Steganography.
- Sending message to public server for intended receipt.
- Confidential communication and secret data storing.

1.5 Objectives and Scope of the project

1.5.1 Objectives

- Develop a Deep Neural Network model for secured Medical Image Steganography.
- Achieve Image Steganography where the PSNR loss will be less than 20%
- Compare the proposed model with recently reported kinds of literature.

1.5.2 Scope of the project

The project is having a scope of hiding images of the medical domain into a natural scene image, and the recipient should be able to decrypt the secret image with PSNR loss of less than 20%

Chapter 2

REQUIREMENT ANALYSIS

Requirement analysis is the process of defining the user's expectation for a model which is to be built. The purpose of requirements analysis is to structure the system independent of the implementation environment.

2.1 Functional Requirements

The way in which system should perform its functionality is basically known as functional requirements. It shows how the system functionalities will behave based on some conditions.

- Input the secrete image which needs to hide and also the cover image to generate a stego image.
- The model shall be able to make both input images to be of the same resolution.
- The model shall be able to decode the encoded secret message.

2.2 Non Functional Requirements

Basically Non functional requirements are the ones that will tell how system shall work and also tells about the system behavior as well as the limitations of the system.

- Model should be robust.
- Percentage of securing data should be able to achieve atleast 85%.
- Input images size must be less than 3MB.

2.3 Hardware Requirements

- Core i5 7th gen processor is required.
- 64 bit Operating System.
- 8GB RAM is preferred.
- GPU is preferred.

2.4 Software Requirements

- Python
- Tensorflow and Keras

Chapter 3

SYSTEM DESIGN

System design is an overview of our model how designed and it provides a conceptual idea about our methodology. it acts as a bridge between our problem statement and the way we are proposing the model.

3.1 System Design

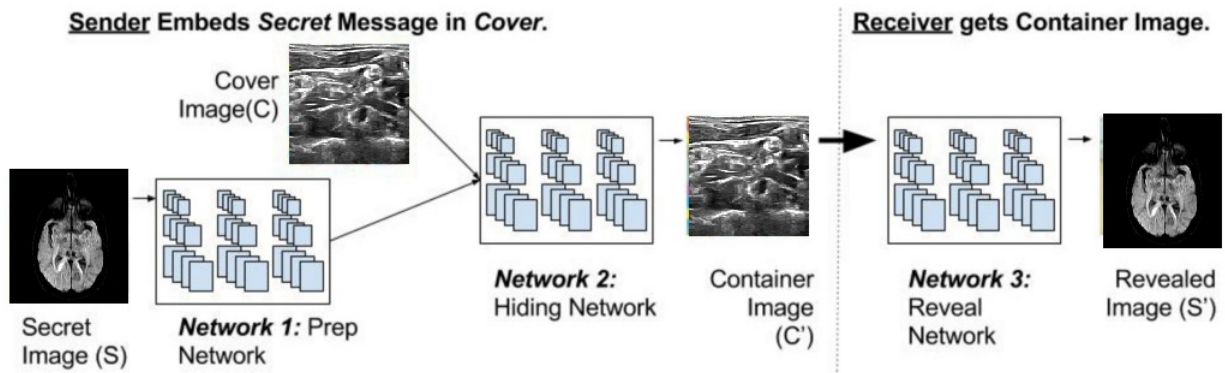


Figure 3.1: System Model

The figure 3.1 shows the system model of this project which contains 3 network layers that perform their respective tasks. The first network is the preparation network where a secret image is taken as input and feed into the model. Next is the hiding network where the cover image is taken as an input and preprocessed. Here the secret image is hidden into the cover image and stego image is generated. This stego image is passed to the revealed network which is the third and last network of the system where decoding of stego image is carried out and decoded secret image is achieved.

3.2 Architectural Design

Figure 3.2 shows the architectural diagram of our model and it describes layers, channels, and filter size used. So our model consists mainly of three networks each have different layers and channels as well as filter size. The Preparation network consists of 2 layers with 65 channels with kernel size 3,4,5 respectively and the Hiding and Reveal network consists of 5 layers with 65 channels with kernel size 3,4,5 respectively.

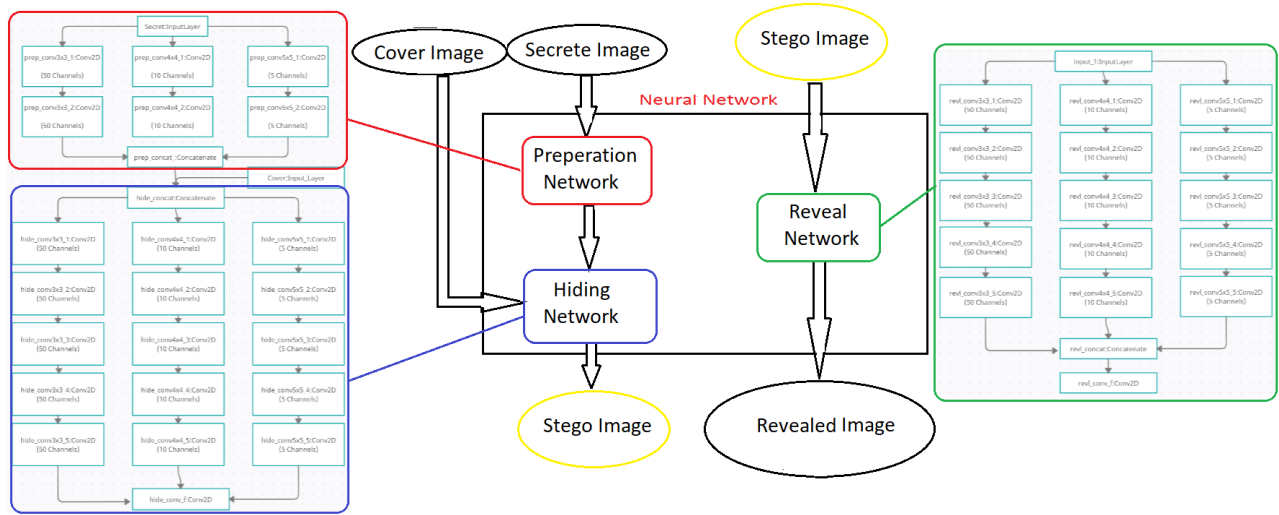


Figure 3.2: Architecture Diagram

3.3 Design Principles

3.3.1 Design Principles 1

Generating deep neural network model consisting of three sub-networks namely preparation network, hiding network, revealing network.

In figure 3.3 preparation network is the first network that takes a secret image as input, preprocesses it, and then passes it to the hiding network. The hiding network is the second network that takes a cover image and secret image as input and generates stego image. This stego image is passed to reveal the network. In reveal network decoding of stego image is carried out to reveal the secret message.

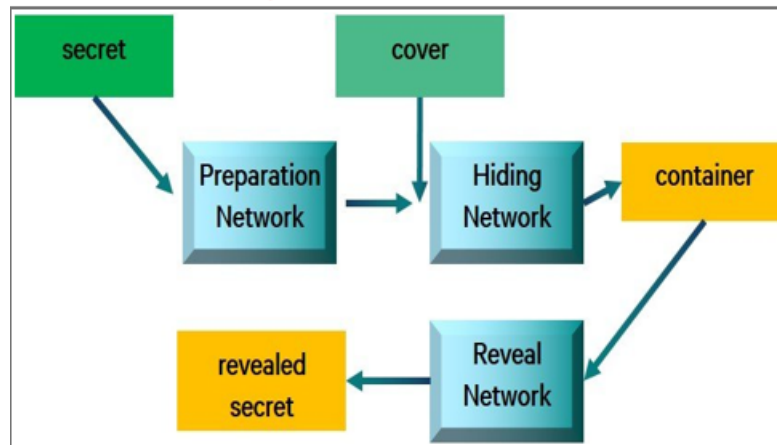


Figure 3.3: CNN Model Generation

3.3.2 Design Principles 2

For building a model a data preprocessing takes longer or more effort to change with wider impact because it holds the data and that could be used by other processes in the system. On the other hand once input, intermediate, outputs have defined the skeleton of the data preprocessing will be clear. This could easily lead to building a successful project.

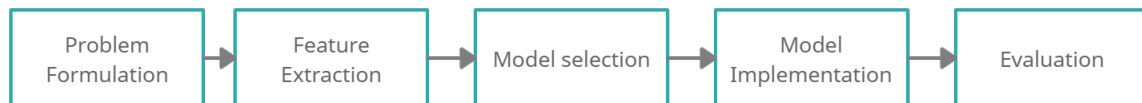


Figure 3.4: Data Preprocessing

3.4 Basic Methodologies design principle

3.4.1 Least Significant Bit (LSB) Steganography

Figure 3.5 Basically any digital image is a collection of some digital values known as pixels. Pixels are the smallest units of an image. Here least significant bit is a method where each pixel's last bit is altered and replaced with the secret image data bits. Each pixel has 3 values namely Red, Green and Blue which all range from 0 to 255 i.e. they are 8-bit values. In this steganography the least significant bits of cover image is taken to conceal the secret image. As we know an 8-bit image is the one in which each pixel is stored as a byte which represents a grayscale value. Suppose if three pixels of cover image have the following values: [11 00 1001 11011110 11101001] And that of secret image is : [11100101 10110110 11110001]. Initially we remove the last 4 bits of cover image by multiplying each pixel with [11110000]. The secret

images pixels are shifted by 4 bits towards right and the adding with cover image pixel. Last 4 LSB of cover image gets replaced by the first 4 MSB of secret image and we get stego image whose first three pixels are [11001110 11011011 11101111]

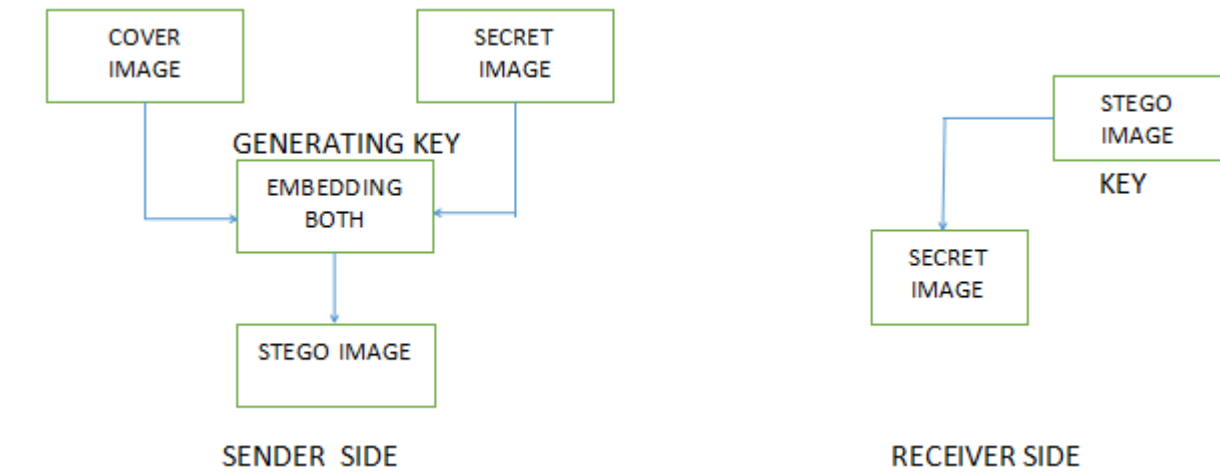


Figure 3.5: LSB

3.4.2 Discrete cosine transform

Figure 3.6 this method is employed to rework the image into the frequency domain. Basically, DCT is implemented in the frequency domain during which the stego-image is transformed from the spatial domain to the frequency domain and every bit is inserted into the frequency components of the cover image. The mathematical functions DCT are applied to hold the digital image data from its spatial domain to the frequency domain. In DCT, after transforming the image within the frequency domain, the info is embedded inside the smallest amount significant little bit of the MF components. DCT method divides the image into parts of differing importance. It transforms the image or a sign from the spatial domain to the frequency domain. It could discrete the image into high, middle, and low-frequency components, in low-frequency sub-band, tons of the signal energy lies at low frequency which contains most vital visual parts of the image while in the high-frequency sub-band, high-frequency components of the image are usually eliminated via compression and noise attacks. therefore the secret message is embedded by adjusting the coefficients of the center frequency sub-band in order that the visibility of the image won't be affected.

3.4.3 Discrete wavelet transform

The WT(Wavelet Transform) is a basic steganography technique helps to describe a multi-resolution decomposition process in terms of expanding of an image into a set of wavelet basis

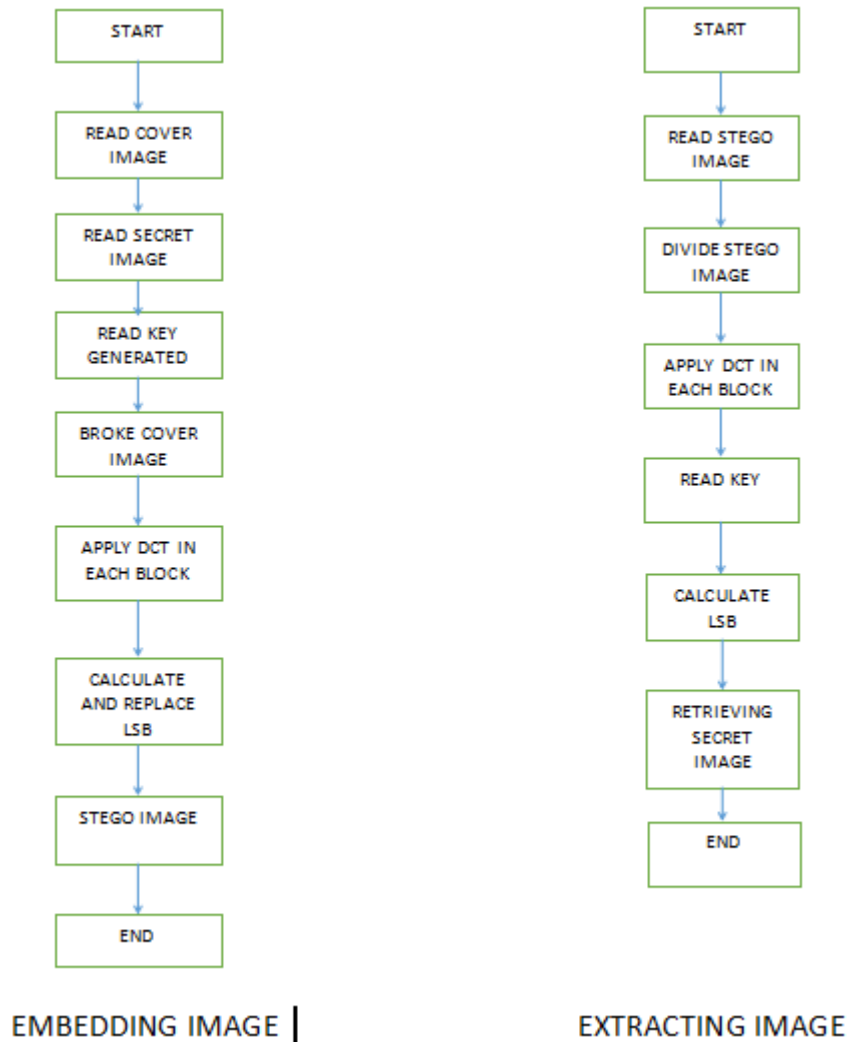


Figure 3.6: Flowchart of Discrete cosine transform

functions based on certain criteria like frequency, light etc. The Discrete Wavelet Transform (DWT) will split the signal into two frequency parts i.e. high and low frequency parts. The high frequency (HF) part contains information about the edge components, while the low frequency (LF) part is split to again into high and low frequency parts. The high frequency components are usually used for steganography methods to hide secret image inside cover image so that human can see the drastic changes in edges of images. Discrete wavelet transform is mainly used in digital images. By applying Discrete wavelet transform, the image is decomposed or divided into four sub-bands: LL, LH, HL and HH. The LL part contains most very important details regarding the images. Embedding in LL part makes stego image more resistant to various attacks by the thief to find secret information hidden in cover image but can lead to distortions in stego image. Decomposition of the light signal into different frequency bands such as LL, LH, HL and HH by discrete wavelet transform which closely matches with

the human visual system. The high frequency sub bands in Discrete Wavelet Transform point the image characteristics such as edges and texture round regions, which are less sensitive to Human visual System characteristics and hence can be used for embedding of secret image inside cover image and outputs a new image called stego image.

Figure 3.7 shows the DWT working at sender side and Figure 3.8 shows the DWT working at receiver side.

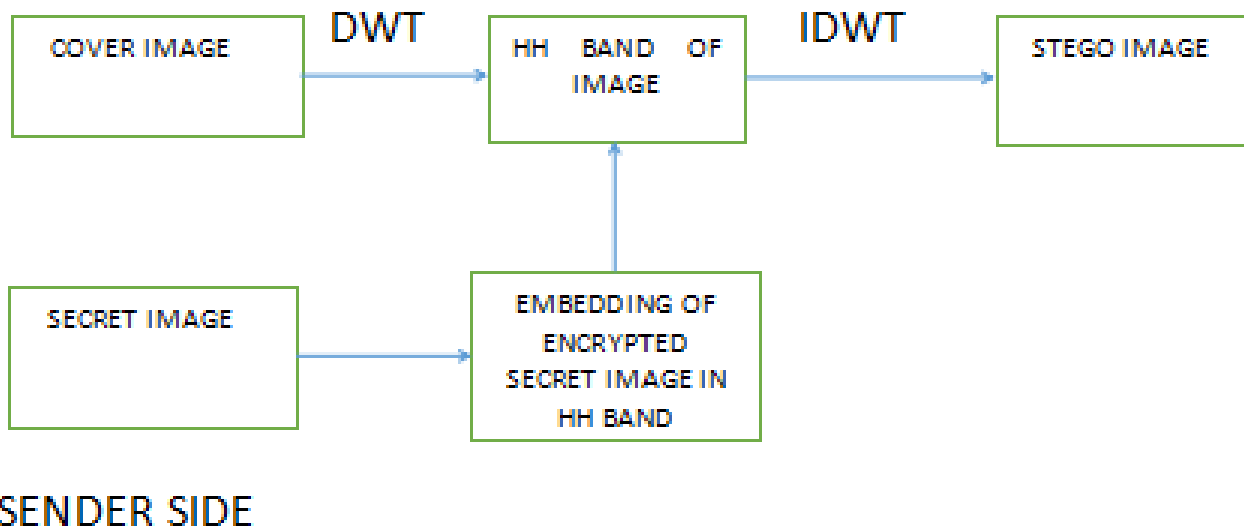
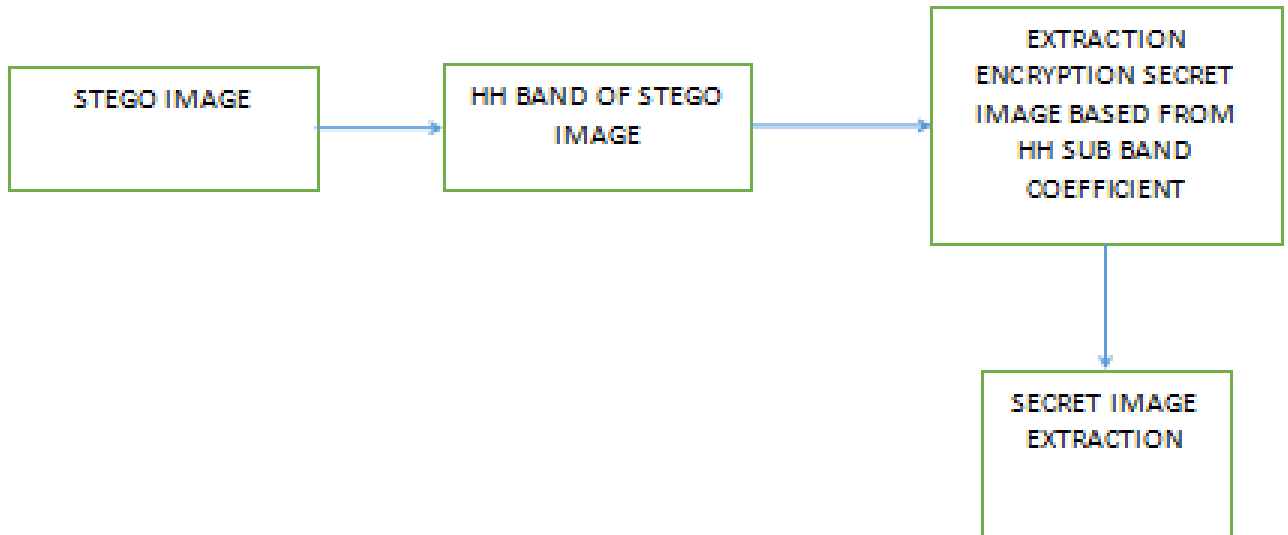


Figure 3.7: Flowchart of Discrete wavelet transform at sender side

3.4.4 Binary pattern complexity

Figure 3.8 Confidential data can be hidden through Digital steganography that means the important information contains inside the secret image is very securely by embedding them into some media data called container image or cover image also named to be Vassel Data. In BPCS-steganography true color images that is 24-bit color images are mostly used for vessel data or container data. Basically to replace the "complex areas" on the bit planes of the vessel image with the confidential data of secret image, the embedding operation is practiced. The very big advantage of BPCS-steganography is that embedding capacity of container image to be very large. To the comparison of simple image based steganography which generally uses the least significant bit of data, and thus only $1/8$ of the total size of the vessel data can be embedded from a 24-bit color image, also BPCS-steganography uses multiple bit planes, and so much higher amount of secret information can be embedded, though this is dependent on the individual image of the input. For a normal image, generally only half of the data might get replaced with secret information before image degradation becomes to be apparent. Principle



RECIEVER SIDE

Figure 3.8: Flowchart of Discrete wavelet transform at reciver side

of the binary pattern complexity is the human visual mechanism such a special property that a too-complicated visual pattern can not be entertained as shape informative. whenever human looks carefully, two same looking areas are entirely different in their sand particle shapes of the required area. This steganography technique makes use of that property. Complexed areas on the bit planes of the vessel data with other complex data patterns that is secret information images gets replaced. Embedding can be reffered as replacing operation. No one can see any difference between the two images of before and after the embedding operation in cover image and stego image. Generally issues arise when the data to be embedded inserted visually as good simple information, if the complex information gets replaced with simple information in the stego or cover image it may create unwanted real image information. In this case the information is passed through a binary image steganography transformation, so that it gets created to a reciprocal complex representation of the embedded process.

3.5 Data structure used

- Tuples

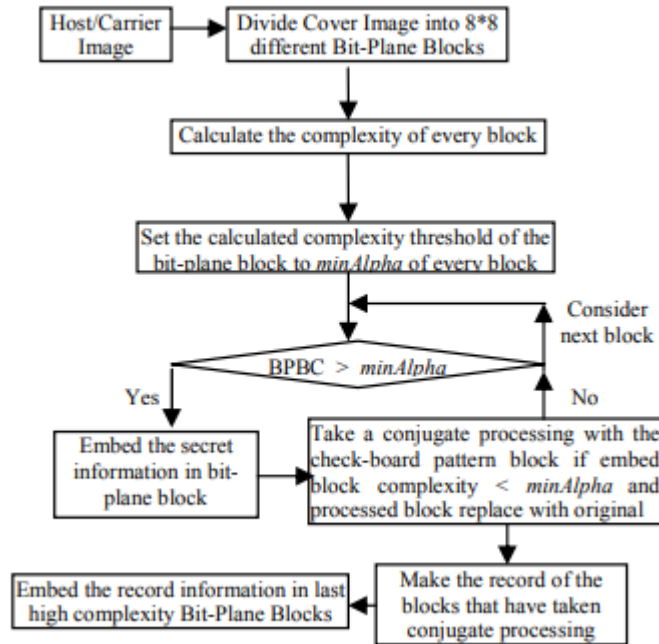


Figure 3.9: Flowchart of BPCS

- List
- Array

3.6 Dataset Description

- Dataset Source
 - We have collected Medical Image dataset of COVID-19 Chest X-ray.
 - Also, we have African Wildlife image dataset which has 4 classes containing total 1504 images.
 - Medical image datasets of COVID-19 Chest X-ray 2,639 images of almost 1GB.
- Dataset Suitability
 - The Medical Image dataset that we have collected will be trained and tested as Secrete Image.
 - African Wildlife dataset will be used hide the secrete image.

Chapter 4

IMPLEMENTATION

For implementation, we used Deep Neural Network for embedding and decoding. By using three main networks called Preparation network, Hiding Network and Revealed Network will do both embedding and decoding. The input secret image taken by the Preparation network makes the secret image to be ready for the embedding process, Hiding network take a cover image and prepared secret image as an input and hides the secret image inside the cover image to produce the stego image (a cover image which embeds secret image inside it) and for decoding it will take input as stego image finds the featured learned by the neural network and reveals the secret image which was embedded in the cover image earlier.

4.1 Proposed Algorithm

Algorithm 1 Embedding and Extracting Process of DeepSteganography

- 1: **Input:** Cover image(c), Secret image(s)
 - 2: Convert the images to vector
 - 3: Normalize the image vector
 - 4: Initialize the convolutional operation weight
 - 5: $x \leftarrow \text{prep}(s)$
 - 6: **for** each in epoch **do**
 - 7: $c' \leftarrow \text{encode}(x, c)$
 - 8: $s' \leftarrow \text{decode}(c')$
 - 9: **end for**
 - 10: $L' \leftarrow$ Compute loss of Model
 - 11: Update the hiding and reveal network weights through computed loss
 - 12: Calculate evaluation metric
 - 13: **Output:** Cover image(c), Secret image(s), Stego image(c'), Revealed image(s')
-

Algorithm 1 shows the implementation details of embedding and extracting process of steganography. It receives two input images Secret and Cover image, each image will be converted into vector once it is vectorized further process to be normalize the image vector. Initialization of weights is done to the Neural Network. For every epoch it will train for encoding and decoding between preparation, hiding and reveal network. Preparation network contain 2 convolutional layers followed by 65 channels per each layer, Hiding network contain 5 convolutional layers with 65 channels per each and Reveal network contains 5 convolutional

network with 65 channels per each. Once the training is done calculated the loss of a model and analysis of performance will be taken care. Output this neural network is extracted secret image from the secret image.

4.2 Data Preprocessing

This is the initial step of model building where the images are resized and pushed into the vector to make the data ready for training and testing.

```

1  import glob
2  from PIL import Image
3
4  #Retriving all image names and it's path with .jpg extension from given directory path in imageNames list
5  imageNames = glob.glob(r"E:/6th sem/dataset/test/*.jpeg")
6
7  #Defining width and height of image
8  new_width = 256
9  new_height = 256
10
11 #Count variable to show the progress of image resized
12 count=0
13
14 #Creating for loop to take one image from imageNames list and resize
15 for i in imageNames:
16     →#opening image for editing
17     →img = Image.open(i)
18     →#using resize() to resize image
19     →img = img.resize((new_width, new_height), Image.ANTIALIAS)
20     →#save() to save image at given path and count is the name of image eg. first image name will be 0.jpg
21     →img.save(r"E:/6th sem/dataset/test/test1/"+str(count)+".jpg")
22     →#incrementing count value
23     →count+=1
24     →#showing image resize progress
25     →print("Images Resized " +str(count)+"/"+str(len(imageNames)),end="\r")

```

Figure 4.1: Data Preprocessing Snippet

4.3 Generation of Deep Neural Network

4.3.1 Preparation Network

The figure of 4.2 preparation network performs two purposes. This network progressively increases the size of the secret image to the size of the cover image for states in which the

secret image($M \times M$) is smaller than cover image($N \times N$), which leads to distributing of secret image's bits across the entire $N \times N$ pixels. The more significant purpose is to transform the color based pixels to more useful features (like edges of the secret message) for encoding the image.

```

1 # encoder, composed by Preparation and Hiding Networks.
2 def make_encoder(input_size):
3     input_S = Input(shape=(input_size))
4     input_C = Input(shape=(input_size))
5
6     # Preparation Network
7     x3 = Conv2D(50, (3, 3), strides = (1, 1), padding='same', activation='relu', name='conv_prep0_3x3')(input_S)
8     x4 = Conv2D(10, (4, 4), strides = (1, 1), padding='same', activation='relu', name='conv_prep0_4x4')(input_S)
9     x5 = Conv2D(5, (5, 5), strides = (1, 1), padding='same', activation='relu', name='conv_prep0_5x5')(input_S)
10    x = concatenate([x3, x4, x5])
11
12    x3 = Conv2D(50, (3, 3), strides = (1, 1), padding='same', activation='relu', name='conv_prep1_3x3')(x)
13    x4 = Conv2D(10, (4, 4), strides = (1, 1), padding='same', activation='relu', name='conv_prep1_4x4')(x)
14    x5 = Conv2D(5, (5, 5), strides = (1, 1), padding='same', activation='relu', name='conv_prep1_5x5')(x)
15    x = concatenate([x3, x4, x5])
16
17    x = concatenate([input_C, x])
18

```

Figure 4.2: Preparation Network Snippet

4.3.2 Hiding Network

The output of the preparation network is taken as input to hiding network and the cover image to create the stego image(container image).

4.3.3 Reveal Network

Figure 4.4 this is the last network i.e. reveal network which is used by the receiver of the image. It is basically the decoder. It receives only the stego image(not the cover image nor secret image). This network extracts the secret information from the stego image and reveals it to the receiver.

4.4 Model Integration

Figure 4.5 the model integration is a step where we merge all the network and returns the result of encoding and decoding.


```

19  # Hiding network
20  x3 = Conv2D(50, (3, 3), strides = (1, 1), padding='same', activation='relu', name='conv_hid0_3x3')(x)
21  x4 = Conv2D(10, (4, 4), strides = (1, 1), padding='same', activation='relu', name='conv_hid0_4x4')(x)
22  x5 = Conv2D(5, (5, 5), strides = (1, 1), padding='same', activation='relu', name='conv_hid0_5x5')(x)
23  x = concatenate([x3, x4, x5])
24
25  x3 = Conv2D(50, (3, 3), strides = (1, 1), padding='same', activation='relu', name='conv_hid1_3x3')(x)
26  x4 = Conv2D(10, (4, 4), strides = (1, 1), padding='same', activation='relu', name='conv_hid1_4x4')(x)
27  x5 = Conv2D(5, (5, 5), strides = (1, 1), padding='same', activation='relu', name='conv_hid1_5x5')(x)
28  x = concatenate([x3, x4, x5])
29
30  x3 = Conv2D(50, (3, 3), strides = (1, 1), padding='same', activation='relu', name='conv_hid2_3x3')(x)
31  x4 = Conv2D(10, (4, 4), strides = (1, 1), padding='same', activation='relu', name='conv_hid2_4x4')(x)
32  x5 = Conv2D(5, (5, 5), strides = (1, 1), padding='same', activation='relu', name='conv_hid2_5x5')(x)
33  x = concatenate([x3, x4, x5])
34
35  x3 = Conv2D(50, (3, 3), strides = (1, 1), padding='same', activation='relu', name='conv_hid3_3x3')(x)
36  x4 = Conv2D(10, (4, 4), strides = (1, 1), padding='same', activation='relu', name='conv_hid3_4x4')(x)
37  x5 = Conv2D(5, (5, 5), strides = (1, 1), padding='same', activation='relu', name='conv_hid3_5x5')(x)
38  x = concatenate([x3, x4, x5])
39
40  x3 = Conv2D(50, (3, 3), strides = (1, 1), padding='same', activation='relu', name='conv_hid4_3x3')(x)
41  x4 = Conv2D(10, (4, 4), strides = (1, 1), padding='same', activation='relu', name='conv_hid4_4x4')(x)
42  x5 = Conv2D(5, (5, 5), strides = (1, 1), padding='same', activation='relu', name='conv_hid5_5x5')(x)
43  x = concatenate([x3, x4, x5])
44
45  output_Cprime = Conv2D(3, (3, 3), strides = (1, 1), padding='same', activation='relu', name='output_C')(x)
46  return Model(inputs=[input_S, input_C],
47              outputs=output_Cprime,
48              name = 'Encoder')

```

Figure 4.3: Hiding Network Snippet

4.5 Loss Function

Figure 4.6 loss function determines the total loss of the model after training where $Loss = |c - c'| + \beta |s - s'|$ where c represent cover image, c' represents stego image, s represent secret image and s' represents revealed secret image.

4.6 Loss Graph

This Figure 4.7 is the graphical representation of loss with respect to epoch upon training.

4.7 Secret message error and cover message error

Figure 4.8 this shows the snippet of pixel difference error between cover and stego image , secret and revealed secret image

```

1 # decoder , composed by the Reveal Network
2 def make_decoder(input_size, fixed=False):
3
4     # Reveal network
5     reveal_input = Input(shape=(input_size))
6
7     # Adding Gaussian noise with 0.01 standard deviation.
8     input_with_noise = GaussianNoise(0.01, name='output_C_noise')(reveal_input)
9
10    x3 = Conv2D(50, (3, 3), strides = (1, 1), padding='same', activation='relu', name='conv_rev0_3x3')(input_with_noise)
11    x4 = Conv2D(10, (4, 4), strides = (1, 1), padding='same', activation='relu', name='conv_rev0_4x4')(input_with_noise)
12    x5 = Conv2D(5, (5, 5), strides = (1, 1), padding='same', activation='relu', name='conv_rev0_5x5')(input_with_noise)
13    x = concatenate([x3, x4, x5])
14
15    x3 = Conv2D(50, (3, 3), strides = (1, 1), padding='same', activation='relu', name='conv_rev1_3x3')(x)
16    x4 = Conv2D(10, (4, 4), strides = (1, 1), padding='same', activation='relu', name='conv_rev1_4x4')(x)
17    x5 = Conv2D(5, (5, 5), strides = (1, 1), padding='same', activation='relu', name='conv_rev1_5x5')(x)
18    x = concatenate([x3, x4, x5])
19
20    x3 = Conv2D(50, (3, 3), strides = (1, 1), padding='same', activation='relu', name='conv_rev2_3x3')(x)
21    x4 = Conv2D(10, (4, 4), strides = (1, 1), padding='same', activation='relu', name='conv_rev2_4x4')(x)
22    x5 = Conv2D(5, (5, 5), strides = (1, 1), padding='same', activation='relu', name='conv_rev2_5x5')(x)
23    x = concatenate([x3, x4, x5])
24
25    x3 = Conv2D(50, (3, 3), strides = (1, 1), padding='same', activation='relu', name='conv_rev3_3x3')(x)
26    x4 = Conv2D(10, (4, 4), strides = (1, 1), padding='same', activation='relu', name='conv_rev3_4x4')(x)
27    x5 = Conv2D(5, (5, 5), strides = (1, 1), padding='same', activation='relu', name='conv_rev3_5x5')(x)
28    x = concatenate([x3, x4, x5])
29
30    x3 = Conv2D(50, (3, 3), strides = (1, 1), padding='same', activation='relu', name='conv_rev4_3x3')(x)
31    x4 = Conv2D(10, (4, 4), strides = (1, 1), padding='same', activation='relu', name='conv_rev4_4x4')(x)
32    x5 = Conv2D(5, (5, 5), strides = (1, 1), padding='same', activation='relu', name='conv_rev5_5x5')(x)
33    x = concatenate([x3, x4, x5])
34    output_Sprime = Conv2D(3, (3, 3), strides = (1, 1), padding='same', activation='relu', name='output_S')(x)

```

Figure 4.4: Reveal Network Snippet

4.8 Histogram Generation

This 4.9 shows the graphical representation of number of occurrence v/s pixel difference between every pair of two consecutive pixels.

4.9 PSNR Calculation

Figure 4.10 PSNR (peak signal noise ratio) is evaluation metric ratio that is used as a quality measurement between two images the one is compressed the original image and another is its compressed image. PSNR is directly proportional to the quality of reconstructed image i.e. higher the PSNR value better the compressed reconstructed image quality.

```

1 # Full model.
2 def make_model(input_size):
3     input_S = Input(shape=(input_size))
4     input_C= Input(shape=(input_size))
5
6     encoder = make_encoder(input_size)
7
8     decoder = make_decoder(input_size)
9     decoder.compile(optimizer='adam', loss=rev_loss)
10    decoder.trainable = False
11
12    output_Cprime = encoder([input_S, input_C])
13    output_Sprime = decoder(output_Cprime)
14
15    autoencoder = Model(inputs=[input_S, input_C],
16                        outputs=concatenate([output_Sprime, output_Cprime]))
17    autoencoder.compile(optimizer='adam', loss=full_loss)
18
19    return encoder, decoder, autoencoder

```

Figure 4.5: Model Integration Snippet

4.10 SSIM Calculation

Figure 4.11 the SSIM which stands for Structural Similarity Index is a evaluation parameter that measures quality of image degradation due to processing like data compressing or by losses in data transmission. It is a full perceptual metric that requires two images that looks similar like a original image and a stego image incase of cover image same for secret image too. The processed image is typically compressed and may loose slight percentage of original information. 'SSIM' is well known in the video and photo industry, but leads to strong application in various fields.

4.11 Correlation Calculation

Figure 4.12 Correlation is a statistical evaluation metric that is used to measure how two images are linearly related. It is used to quantify simple relationship without making any statement about effect and cause.

4.12 Intersection Calculation

Figure 4.13 An intersection is an junction where more numbers of features can converge,

```

1 # Variable used to weight the losses of the secret and cover images (See paper for more details)
2 beta = 1.0
3
4 def rev_loss(s_true, s_pred):
5     # Loss for reveal network is: beta * |S-S'|
6
7     print(s_true.shape, s_pred.shape)
8     return beta * K.sum(K.square(s_true - s_pred))
9
10
11 # Loss for the full model, used for preparation and hiding networks
12 def full_loss(y_true, y_pred):
13     # Loss for the full model is: |C-C'| + beta * |S-S'|
14     s_true, c_true = y_true[...,0:3], y_true[...,3:6]
15     s_pred, c_pred = y_pred[...,0:3], y_pred[...,3:6]
16
17     #s_loss = rev_loss(s_true, s_pred)
18     s_loss = beta * K.sum(K.square(s_true - s_pred))
19     c_loss = K.sum(K.square(c_true - c_pred))
20
21     return s_loss + c_loss
22

```

Figure 4.6: Loss Function Snippet

diverge, meet or cross. Intersections are often delineated by common features and may be classified by feature segments.

4.13 KL Divergence

Figure 4.14 Kullback-Leibler Divergence known as KL Divergence refers to quantify the measurement of the difference between 2 probability distribution over same base image. Basically, Kullback-Leibler divergence of $q(x)$ from $p(x)$, denoted by $DKL(p(x), q(x))$, is the quantitative measure of the image quality data lost when $q(x)$ is compressed and processed to reach close to $p(x)$.

```

1 # Plot loss through epochs
2 plt.plot(loss_history)
3 plt.title('Model loss')
4 plt.ylabel('Loss')
5 plt.xlabel('Epoch')
6 plt.show()

```

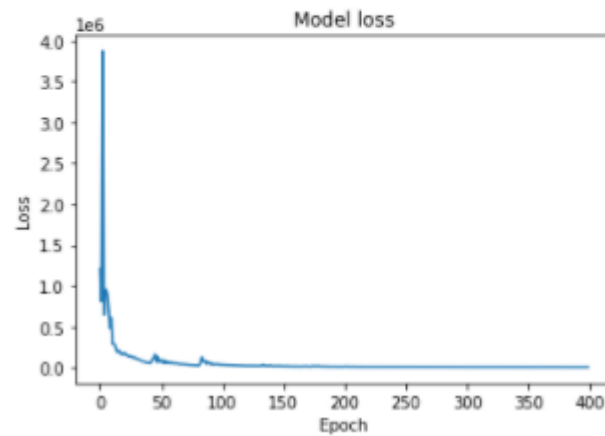


Figure 4.7: Loss Graph

```

[] 1 # Print pixel-wise average errors in a 256 scale.
2 S_error, C_error = pixel_errors(input_S, input_C, decoded_S, decoded_C)
3
4 print ("S error per pixel [0, 255]:", S_error)
5 print ("C error per pixel [0, 255]:", C_error)

```

```

S error per pixel [0, 255]: 10.212219
C error per pixel [0, 255]: 11.466626

```

Figure 4.8: Pixel-wise average error Snippet

```
[ ] 1 import cv2
    2 import numpy as np
    3 from matplotlib import pyplot as plt
    4
    5 img1 = cv2.imread('/content/drive/MyDrive/Evaluation_Metric/secret7.png', -1)
    6 img2 = cv2.imread('/content/drive/MyDrive/Evaluation_Metric/decrypted7.png', -1)
    7 fig, ax = plt.subplots()
    8 histr = cv2.calcHist([img1],[2],None,[256],[0,256])
    9 histr1 = cv2.calcHist([img2],[2],None,[256],[0,256])
   10 plt.plot(histr,color = 'b')
   11 plt.plot(histr1,color = 'r')
   12 plt.xlim([0,256])
   13 plt.ylim([0,1500])
   14 ax.legend(['Cover Image', 'Stego Image'], loc='upper left')
   15 plt.title('Pixel Difference Histogram of Cover and Stego image ')
   16 plt.xlabel('Pixel Difference between every pair of two consecutive pixels')
   17 plt.ylabel('Number of occurrence')
   18 metric_val = cv2.compareHist(histr, histr1, cv2.HISTCMP_CORREL)
   19 plt.show()
```

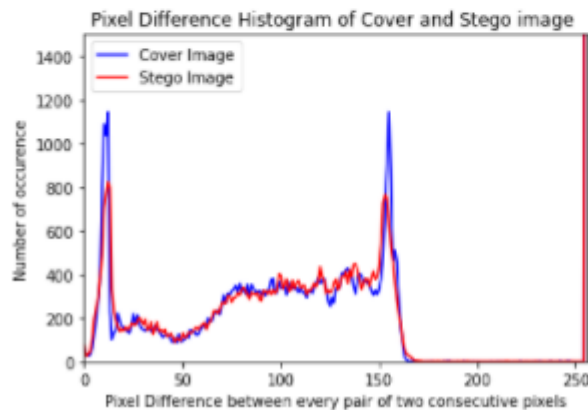


Figure 4.9: Histogram Generation Snippet

▼ PSNR

```
[ ] 1 import numpy
    2 import math
    3 import cv2
    4 original = cv2.imread("/content/drive/MyDrive/Evaluation_Metric/secret7.png")
    5 contrast = cv2.imread("/content/drive/MyDrive/Evaluation_Metric/decrypted7.png")
    6 def psnr(img1, img2):
    7     mse = numpy.mean( (img1 - img2) ** 2 )
    8     mse=mse/2
    9     if mse == 0:
   10         return 100
   11     PIXEL_MAX = 255.0
   12     return 20 * math.log10(PIXEL_MAX / math.sqrt(mse))
   13
   14 d=psnr(original,contrast)
   15 print("PSNR: ", d)
   16
   17 loss=(48-d)*100/48
   18 print("PSNR Loss=",loss)
```

```
PSNR: 41.78709625740858
PSNR Loss= 12.943549463732131
```

Figure 4.10: PSNR Calculation Snippet

▼ SSIM

```
[ ] 1 from skimage.measure import compare_ssim
    2 import argparse
    3 import imutils
    4 import cv2
    5 # 3. Load the two input images
    6 imageA = cv2.imread("/content/drive/MyDrive/Evaluation_Metric/secret7.png")
    7 imageB = cv2.imread("/content/drive/MyDrive/Evaluation_Metric/decrypted7.png")
    8
    9 # 4. Convert the images to grayscale
   10 grayA = cv2.cvtColor(imageA, cv2.COLOR_BGR2GRAY)
   11 grayB = cv2.cvtColor(imageB, cv2.COLOR_BGR2GRAY)
   12
   13 # 5. Compute the Structural Similarity Index (SSIM) between the two
   14 #   images, ensuring that the difference image is returned
   15 (score, diff) = compare_ssim(grayA, grayB, full=True)
   16 diff = (diff * 255).astype("uint8")
   17
   18 # 6. You can print only the score if you want
   19 print("SSIM: {}".format(score))
```

SSIM: 0.9579188124837831

Figure 4.11: SSIM Calculation Snippet

▼ CORRELATION

```
[ ] 1 import cv2
2 img1 = cv2.imread('/content/drive/MyDrive/Evaluation_Metric/cover7.png')
3 img2 = cv2.imread('/content/drive/MyDrive/Evaluation_Metric/stego7.png')
4
5 # Convert it to HSV
6 img1_hsv = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)
7 img2_hsv = cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)
8
9 # Calculate the histogram and normalize it
10 hist_img1 = cv2.calcHist([img1_hsv], [0,1], None, [180,256], [0,180,0,256])
11 cv2.normalize(hist_img1, hist_img1, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX);
12 hist_img2 = cv2.calcHist([img2_hsv], [0,1], None, [180,256], [0,180,0,256])
13 cv2.normalize(hist_img2, hist_img2, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX);
14
15 # find the metric value
16 metric_val = cv2.compareHist(hist_img1, hist_img2, cv2.HISTCMP_CORREL )
17
18
19 print(metric_val)
```

0.9346523131797626

Figure 4.12: Correlation Calculation Snippet

▼ INTERSECTION

```
[ ] 1 import cv2
    2 img1 = cv2.imread('/content/drive/MyDrive/Evaluation_Metric/secret7.png')
    3 img2 = cv2.imread('/content/drive/MyDrive/Evaluation_Metric/decrypted7.png')
    4
    5 # Convert it to HSV
    6 img1_hsv = cv2.cvtColor(img1, cv2.COLOR_BGR2HSV)
    7 img2_hsv = cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)
    8
    9 # Calculate the histogram and normalize it
   10 hist_img1 = cv2.calcHist([img1_hsv], [0,1], None, [180,256], [0,180,0,256])
   11 cv2.normalize(hist_img1, hist_img1, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX);
   12 hist_img2 = cv2.calcHist([img2_hsv], [0,1], None, [180,256], [0,180,0,256])
   13 cv2.normalize(hist_img2, hist_img2, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX);
   14
   15 # find the metric value
   16 metric_val = cv2.compareHist(hist_img1, hist_img2, cv2.HISTCMP_INTERSECT )
   17
   18
   19 print(metric_val)

0.9223684853284808
```

Figure 4.13: Intersection Calculation Snippet

```
from keras.preprocessing import image

img_i = image.load_img("/content/sample_data/decrypted1.png")
img_d= image.load_img("/content/sample_data/secret1.png")

x=image.img_to_array(img_i)
y=image.img_to_array(img_d)

import numpy as np
p = np.asarray(x, dtype=np.float)
q= np.asarray(y, dtype=np.float)

k=np.sum(np.where(p != 0, p * np.log(p / q)/q, 0))
```









Figure 4.14: KL Divergence Snippet

Chapter 5

RESULTS AND DISCUSSIONS

This chapter describes about result obtained and comparative analysis with state of the art. To get the optimal result, we experimented our model on different inputs and evaluated as per the following quantitative evaluation metric. The proposed model has been compared with various papers published recently.

5.1 Basic Method Results

Sl. No.	Basic Methods	Cover Image	Secret Image	Stego Image	Decoded Image	PSNR (In db)	PSNR Loss (In %)
1.	LSB					36.20	24.58
2.	DCT					31.44	34.5

3.	DWT					22.81	52.41
4.	BPS					37.87	29.43

Figure 5.1: Implemetation results of LSB,DCT,SWT,BPS

5.2 Comparative Analysis

Expt No.	PSNR		PSNR		SSIM	KL-Divergence	Intersectionin	Correlation
	Cover vs Stego	PSNR Loss	Secrete vs De-coded	PSNR Loss				
1	33.23	30.77	38.66	19.46	0.93	556.24	0.96	0.97
2	34.14	28.88	39.4	17.92	0.93	910.35	0.94	0.98
3	35.39	26.27	41.04	14.50	0.97	1962.29	0.97	0.97
4	34.73	27.65	40.84	14.92	0.95	700.54	0.95	0.98
5	33.85	29.48	38.27	20.27	0.92	220.48	0.95	0.96
6	35.61	25.81	39.99	15.69	0.92	1763.29	0.94	0.94
7	36.18	24.63	41.78	12.96	0.96	1367.22	0.97	0.98
8	33.77	29.65	39.49	19.81	0.93	322.54	0.94	0.95
9	33.45	30.31	37.84	21.17	0.93	1408.14	0.95	0.95
10	35.93	25.15	39.52	17.67	0.95	367.06	0.97	0.97
Avg	34.63	27.86	39.58	17.54	0.94	957.82	0.95	0.97

Expt No.	Author	Prop. Method	PSNR	SSIM	KL-Div	Pixel Error Cover	Pixel Error Secret
1	Sumit Balugat, 2020	Deep neural networks are concurrently trained so as to develop hiding and revealing network and are designed to work as together. The model was trained on images taken from ImageNet dataset and also worked with variety of sources.	32.87	0.87	857.32	10.32	11.58
2	Vijay Kumar 2019	The encoder neural network determines where the hiding message has to be places and distribute allover the bits of cover image. Recieving side will have decoder network that concurrently trained along with the encoder network ,which will extract the secret image. In this model the main work come out to be producing minimum distortion for the extracted secret message or data. Thus ,can maintain its solidarity.	-	-	-	4.30	4.68
3	Yueyun Shang 2019	This methodology presents one way conduction of deep learning network in space. They proposed a basic/novel steganography method which strengthen the security.	27.23	-	-	-	-
4	Ching-Chun Chang 2018	Deep learning has brought about a phenomenal paradigm shift in digital steganography.	34.29	-	-	-	-
5	Kartik Sharma 2017	Combines steganography and cryptography with neural network to hide the secret image in another image of any size.	-	-	-	13.81	11.29
6	Prop. method	DNN based Image Steganography.	30.82	0.86	913.79	14.17	13.75

Table 5.1: Comparison of results with existed works






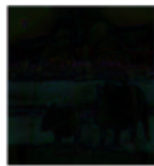
Cover Image	Secret Image	Stego Image	Decoded Image	Diff Cover	Diff Secret
					

Figure 5.2: Proposed Methodology Output

Chapter 6

CONCLUSION AND FUTURE SCOPE OF THE WORK

6.1 Conclusion

We have presented a deep learning approach for deep steganography and proposed a method that has a slightly better performance than traditional method. The proposed methodology has been compared with recently published papers with respect to various evaluation metric like PSNR,SSIM,KL divergence.Firstly developed a system/model that consists of three sub network using deep neural network in order to hide the secret message/data within another image called cover image or container image.Instilled a color image(secret image) in another image of same resolution devoting more amount of bits to the secret image. Although, by compression accomplished by deep-nueral networks,the extracted results of secret image, and also the appearence of the stego image which contains the hidden image, closely estimated the original secret and stego images respectively.

6.2 Future Scope

The model can be optimized further by training it on larger dataset and using high performance system like GPU. The model can be further updated to hide multiple images in a single image.

REFERENCES

- [1] Sara T Kamal, Khalid M Hosny, Taha M Elgindy, Mohamed M Darwish, and Mostafa M Fouda. A new image encryption algorithm for grey and color medical images. *IEEE Access*, 9:37855–37865, 2021.
- [2] Muhammad Arslan Usman and Muhammad Rehan Usman. Using image steganography for providing enhanced medical data security. In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–4. IEEE, 2018.
- [3] K Shankar, Mohamed Elhoseny, E Dhiravida Chelvi, SK Lakshmanaprabu, and Wanqing Wu. An efficient optimal key based chaos function for medical image security. *IEEE Access*, 6:77145–77154, 2018.
- [4] MS Sreekutty and PS Baiju. Security enhancement in image steganography for medical integrity verification system. In *2017 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pages 1–5. IEEE, 2017.
- [5] Hayat Al-Dmour and Ahmed Al-Ani. Quality optimized medical image steganography based on edge detection and hamming code. In *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, pages 1486–1489. IEEE, 2015.
- [6] Shubham Lavania, Palash Sushil Matey, and V Thanikaiselvan. Real-time implementation of steganography in medical images using integer wavelet transform. In *2014 IEEE International Conference on Computational Intelligence and Computing Research*, pages 1–5. IEEE, 2014.