Data Hiding in Images using Steganography

BHOOVIKA V (Roll no : 19Z310)
DHANAVANDHANA K (Roll no : 19Z311)
PAVITHRA YAZHINI G K (Roll no : 19Z337)
PRIYADHARSHINI J (Roll no : 19Z339)
SWETHA M (Roll no : 19Z355)

Project Guide : Dr. V. SANTHI,
PROFESSOR,
DEPT. OF CSE,
PSG COLLEGE OF TECHNOLOGY,
COIMBATORE.

Dissertation submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING
Branch:COMPUTER SCIENCE AND ENGINEERING
Of Anna University

April 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
PSG COLLEGE OF TECHNOLOGY
(Autonomous Institution)
COIMBATORE – 641 004

PSG COLLEGE OF TECHNOLOGY
(Autonomous Institution)
COIMBATORE – 641 004

Data Hiding in Images using Steganography

Bona fide record of work done by

BHOOVIKA V (Roll no : 19Z310)
DHANAVANDHANA K (Roll no : 19Z311)
PAVITHRA YAZHINI GK (Roll no : 19Z337)
PRIYADHARSHINI J (Roll no : 19Z339)
SWETHA M (Roll no : 19Z355)

Dissertation submitted in partial fulfillment of the requirements for the degree

of

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE AND ENGINEERING

of Anna University

April 2023

……	……..
Dr.V.Santhi	Dr.G.Sudha Sadasivam
Faculty Guide	Head of the Department

Certified that the candidate was examined in the viva voce examination held on ……

……	……
(Internal Examiner)	(External Examiner)

CERTIFICATE

Certified that the report titled **“Data Hiding in Images using Steganography"** for the project work-II(19Z820) is a genuine work of of **Bhoovika V (19Z310), Dhanavandhana K(19Z311), Pavithra Yazhini GK (19Z337), Priyadharshini J (19Z339)** and **Swetha M (19Z355)** who have performed the task under my direction for the partial completion of the standards required for the Bachelor of Engineering degree in Computer Science and Engineering. Furthermore, I certify that, to the best of my knowledge and belief, the work reported here does not comprise a portion of any other thesis or dissertation that served as the foundation for a prior degree or award.

Place : Coimbatore
Date :

Dr. V. Santhi
Professor
Department of Computer Science and Engineering
PSG College of Technology
Coimbatore-641004

COUNTERSIGNED

HEAD
Department of Computer Science and Engineering
PSG College of Technology,
Coimbatore-641004

ACKNOWLEDGEMENT

The Project work-II Laboratory we had, gave us an irreplaceable platform to enhance our skills and knowledge. We feel privileged to be associated with such a knowledgeable curriculum in **PSG College of Technology**.

We are really thankful that our principal, Dr. K. Prakasan, gave us this chance which has made us drench in the knowledge shower.

We are glad to express our gratitude to our HOD, Professor G. Sudha Sadhasivam from the Department of Computer Science and Engineering, for being the pillar of support.We would want to use this opportunity to place our sincere thanks to our tutor **Ms.K.Vani, Assistant Professor (Sr.Gr) , CSE department** who has shown her support in all the ways for completing our project with ease and also want to thank our guide **Dr.V.Santhi, Professor, CSE department** who has spent her valuable time to aid us for coming out with the good project.

We also wish to extend our heartfelt thanks to my teammates whose inquisitiveness has always been a driving force to learn unique things.

CONTENTS

CHAPTER **Page No.**

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Data security requires the use of steganography, an approach for data concealment. Then to keep the existence of the data a secret, it tries to cover up important data using a cover medium. Only the sender or the intended recipient will be aware of the secret information because it is concealed. Using techniques that are undetectable to the human eye, image steganography involves hiding data - text, photos, sounds, or videos—within a cover image. The main objectives of steganography are to increase steganographic capabilities, increase imperceptibility, and maintain robustness. Besides that, there really are a variety of methods for image steganography. In this study, we provide an edge detection-based deep learning method for image steganography. Compared to other edge detection methods, the suggested technique can keep more edge pixels. Now to retain more information and improve embedding capacity, we intend to make use of the edges that are present in an image. In other words, by placing more bits of secret image in edge pixels and less bits of secret image in non-edge pixels, we are hiding secret bits within the cover image. Next is to insert the concealed image, We begin by removing feature maps from the cover image by using an edge detection technique and a pre-trained Holistically-Nested edge detection (HED) model.Then we perform embedding and extraction of the secret image. Finally to show that the suggested approach provides a higher payload, PSNR value and SSIM, the performance of the steganographic image is evaluated.

CHAPTER 1

INTRODUCTION

1.1 Introduction

The massive advancement in digitalization has made data communication considerably simpler. Millions of users have therefore shown interest in this for amusement, information sharing, and other relevant activities. Communication through a public channel is always uneasy, though. The main goal of digital data communication is to conceal exclusive data from unauthorized and dishonest interceptors. In order to achieve privacy, cryptography and steganography are crucial. To mislead the opponent, cryptography converts the plaintext into an encipher form. Since interceptors aren't even made aware that the secret data is there, steganography is a different kind of data-hiding technique from cryptography. Steganography hides hidden data using a variety of digital elements, including text, audio, video, and images. The ease of use and regular availability of digital images have led to a sharp improvement in the popularity of image steganography techniques among steganographers. Both the spatial and frequency domains can also be used for image steganography. The embedding of secret data into the pixels of the cover image is directly performed by the spatial domain.

Reversible data data hiding (RDH) and irreversible data hiding (Non-RDH) are two categories for data hiding techniques. Both the cover image and the secret data are fully recovered using the RDH method. The Non-RDH techniques, on the other hand, concentrate on the successful retrieval of only the secret data. Images are frequently used as cover objects in steganography. Based on the secret key, an image is embedded (as the cover picture) into another image using an embedding method. The stego image, which is a byproduct of the embedding process, will be received by the recipient. On the opposite side, an extraction

algorithm processes the stego image. Unauthorized individuals could view the image transmission but not the stego image, nor could they find or decipher the hidden message.

In this project, we attempt to place a color image inside another image. Always size of a secret image is less than the cover image's size. So here we had proposed a deep learning-based technique for image steganography by means of edge detection. And we aim to utilize the edges present in an image to store more data and achieve better embedding capacity. We have used a pre-trained Holistically-Nested edge detection (HED) model to extract the feature maps from cover images and modify them using edge detection techniques to embed the hidden image.

Also, we encrypt the secret message and decrypt them while revealing them for additional security purposes. We not only show that how this deep learning is used to successfully hide images, but we also thoroughly analyze how the result is obtained and consider extensions. And the proposed method performs well on real-world images from many different sources.

1.2 Problem Statement

To hide or embed a secret information (secret image) inside a cover image seamlessly with the help of edge detection method in deep learning where we can hide more data in edges and less data in non edges for achieving better embedding capacity. It helps to extract back the secret information (secret image) from the cover image with minimum loss.And to demonstrate that, in comparison to previous edge detection techniques used in deep learning, the developed edge detection method has a better amount of true edges and a lower noise ratio.

CHAPTER 2

LITERATURE SURVEY

To conceal sensitive information within the cover image, a deep learning-based Steganography technique is presented in [1]. To achieve this, a CNN - convolutional neural network with a Deep Supervision-based edge detector is employed, which can retain more edge pixels than traditional edge detection algorithms. The proposed method's the three main components are pre-processing and edge detection, embedding, and extraction. It is reported that the model employs CNN with Deep Supervision-based edge detection. Initial input is the cover image(CI), and the masked image(MI) is created by masking the final five bits of each image of CI. The reference edge image is then computed from MI using CNN edge detection based on Deep Supervision. Then, to create four distinct reference images(RI), the gray-scale edge image is binarized using Hard thresholding, Otsu thresholding, Adaptive mean thresholding, and Gaussian adaptive thresholding. The stego image(SI) is then obtained by embedding more bits in edge pixels and fewer bits in non-edge pixels. The same pre-processing (i.e., masking) is performed on SI at the receiver's end, and the same edge recognition technique is then used to recreate the same RI. Secret SEI components are derived from RI.

Then in the embedding phase, it is easy to distinguish between edge and non-edge pixels using the recovered RIs. A 1D bitstream named "s" is created from the secret image (SEI). The LSB technique is then applied for embedding 'x' amount of bits from s in each edge pixel of CI and 'y' number of bits in each non-edge pixel of CI. Here, x is always bigger than y because more bits can be embedded in the edge pixels to produce a SI that is less distorted. Additionally, a pixel adjustment procedure is introduced to reduce distortion and improve the SI's visual appeal without impairing the hidden data. Finally at the extraction phase, which SI pixel is an edge pixel and which is in the backdrop can be determined with ease from RI. As a result, SI's edge pixels yield x numbers of LSBs, while a non-edge pixel yields y numbers of LSBs. The bits that were taken are kept in 1D arrays. The SEI can be immediately identified from s. While

maintaining a respectable payload capacity, the PSNR obtained from the suggested method surpasses some existing techniques in both edge-based and spatial steganography methods.

In [2], a brand-new edge detection technique that addresses the multi-scale and multi-level feature learning and holistic picture training and prediction challenges of this age-old vision problem is devised. The proposed method, holistically-nested edge detection (HED), employs a deep learning model that performs image-to-image prediction using fully convolutional neural networks and deeply supervised nets. HED naturally learns complex hierarchical representations by deep supervision on side responses, which is essential for overcoming the challenging ambiguity in edge and object boundary identification. We greatly advance the state-of-the-art on the BSD500 dataset (ODS F-score of.782) and the NYU Depth dataset (ODS F-score of.746), and we do it at a speed (0.4s per image) that is orders of magnitude quicker than other previous CNN-based methods.

The research in [3] describes a new method of steganography called online information concealment on instrument output screens. It uses a private marking system utilizing symmetric key steganography and LSB approach to read the source of the image and block the picture into [R x C] smaller sections. Each part of every block contains one bit of the hidden message, and the rest of the blocks are filled with the other bits in a similar manner. The method allows for the secure transmission of huge amounts of sensitive data between two communication participants, and has numerous point-to-point and multipoint communications practical, private, and military applications.

This study [4] provides a steganographic method of obscuring sensitive information in order to enable safe communication. The Arnold transformation is applied to the selected cover image, and data bits from the hidden image are inserted into the jumbled cover. The image is then transformed in reverse using the Arnold algorithm. The experimental findings support the assertion that the created stego is virtually undetectable, resulting in a stego that is undetectable and does not draw attention from undesirable sources.

In [5], image steganography and RSA are used to safeguard data (asymmetric algorithm). The data is encrypted using the RSA method, and the encoded data is then embedded in the cover image. The cover image for this publication is a 24 bit RGB image, and data is embedded in 4 LSBs of the RGB component of the pixels. In order to create a stego image, the mapping function changes the position of embedded bits from encrypted data at position to a cover image. With a greater PSNR and significantly lower MSE value, RSA using the modified LSB technique is more dependable than RSA using the LSB technique.

This publication [6] proposes an image steganography strategy that uses an image expansion notion and the LSB matching method to produce a high-quality stego image. The mechanism of concealing the secret information within the cover media is called an embedding algorithm, where every pixel on cover image is searched for matching bits and the amount of bits targeted for embedding is defined by the similarity threshold. The extraction algorithm is utilized to obtain the hidden information from stego medium and allows for distortion-free embedding of a significant amount of data. The proposed scheme's capacity will improve if the LSB number of bits is increased. Experimental results suggest that the approach is successful in obtaining a more embedding ability and high-quality image, which is essential for steganographic technologies.

The proposed solution in [7] is to hide text in image using LSB(Least Significant Bit) steganography and AES(Advanced Encryption Standard). The four components of the process are encryption, encoding, decryption, and decoding. AES is to encrypt data, and the user-defined image and cipher text created by the encryption module are inputs to the encoding module. The Python Imaging Library (PIL) is used to create cipher images, and each image pixel's red channel LSB value is swapped out for a cipher image pixel. The encoded steno image is entered into the decoding module, where the concealed data is recovered using image processing methods. Peak Signal to Noise Ratio and Mean Square Error are to examine the distortion and noise in the same image of various sizes.

The Least Significant Bit method is to incorporate a hidden secret message using steganography techniques. This study [8] employs 24 pixels to save the image's size in cover and imply the LSB technique to obscure image. The management position will coordinate the positioning of hidden image's size and pixel values. Images recovery results show MSE and RMSE values of 0, followed by an infinitive for the PSNR value. This demonstrates that the image is the proper secret message and that the photographs are still of high quality.

The science and art of concealing a concealed message in different file types is called image steganography, and this work [9] explores the numerous strategies, techniques, and plans utilized in this field. Steganography falls into the following four categories: carrier object, secret message, and steganographic method. Pixel-Value Differentiating (PVD) scheme uses the difference value between two consecutive pixels in a block to determine how many secret bits should be embedded, while LSB algorithm replaces the most-right bits of a cover file byte, GLM modifies the gray levels of the image pixels, and PVD algorithm modifies the gray levels of the image pixels. Also, a technique for minimizing the difference between the old and new pixel values in a steganographic image was discussed.

In [10] deep neural networks are made use to encode and decode many hidden images inside of a single cover image with the same resolution. The system contains three parts: the Preparation Network, which enlarges the secret image if it is smaller than the cover image, the Hiding Network, which generates the Container image, and the Show Network, which eliminates the cover image to reveal the hidden image. The Tiny ImageNet dataset was used, and noise was added to the original cover image to increase security. The loss for all values is predicted to rise as more image features are hidden in a single image.

This paper [11] discusses the use of encryption and concealment techniques to protect data transfer from intruders. It suggests a different steganography method that contains two procedures in the concealing stage and transforms the image into a byte array during the extraction process. The said method for evaluating it was subjected to MSE and PSNR tests,

yielding good results when assessing the resolution of hidden image and comparing it to the actual picture.

An LSB and Deflate compression method combination is offered for image steganography in this research study [12]. It employs the LSB method and a linked list to insert a secret message in a color image. Prior to embedding, the cover image and concealed message will both be prepared. To evaluate the efficacy of the recommended steganography approach, the two standard metrics of the mean-squared error (MSE) and the peak signal-to-noise ratio (PSNR) will then be taken into account. While PSNR is used to statistically evaluate the quality of an image, MSE is used to calculate the total quadratic error between both the cover image and the steganography image. The suggested methods raised MSE and PSNR scores, proving the benefit of data compression.

Least Significant Bit (LSB) replacement is a technique for data concealment known as steganography that replaces the cover image's LSB with hidden bits. An effective way to increase capacity while taking good visual quality into account is suggested in Paper [13]. The secret data's maximum and lowest values are established by the algorithm, which also utilises a flag to signify inverted bits. All values in the data are then deducted from the maximum value. The quotients and remainders are kept in an array, and the cover image is split into two sections by LSB inversion: one for embedding the quotients and the other for the remainders. The linear time complexity of the LSB embedding algorithm is O. (n). Experimental results show that the current method enhances the quality and embedding capability of the stego picture.

The authors of this paper [14] modified an existing solution of reversible data hiding through their novel proposal of encryption method. The proposed solution has 3 bits of information embedded in an image block of 'b x b' pixels, and requires the sender and receiver to share only three encryption / decryption keys. The first step is hiding data and image encryption, followed by RC4 stream generator, conversion into matrices, repositioning, SVM training, and data extraction and recovery. The proposed method's primary goal was to obtain good data embedding capacity, and it needed to have a low bit error rate in contrast to existing

approaches. The results demonstrated that the suggested strategy took less time to execute and had strong embedding potential.

The authors of [15] performed steganography using the k-least significant bit from cover image, inserting one cover image inside the other. The most significant bit was chosen at the sending side and an entire secret image was concealed inside cover image. The process of stego picture extraction uses the same procedures as the encoding phase, and an algorithm is implemented to look through the many regions where the secret data is hidden. Used a local entropy filter to identify the region that houses the hidden image, and a technique for enhancing image quality is employed to improve the image resolution. PSNR is to assess how well the suggested strategy for integrating images works, and the proposed algorithm results in less distortion and less information loss.

A secret image can be concealed in clear sight by means of image steganography. The cover image is typically statistically altered to embed the secret binary bits after the secret data is converted into binary bits. The cover image may become distorted in a result of being overloaded, making the secret information obvious. As a result, the traditional methods have a limited ability to conceal. To embed a secret image inside a cover image and to extract the embedded secret image from the cover image, a lightweight yet straightforward deep convolutional autoencoder architecture is suggested in this paper [16]. Three datasets—COCO, CelebA, and ImageNet—are used to assess the suggested methodology. Results from the test set's Peak Signal-to-Noise Ratio, Hiding Capacity, and Imperceptibility tests are used to gauge success. The discussed method was tested against other conventional image steganography techniques using a variety of images, including Lena, an airplane, a baboon, and peppers. The experimental results have demonstrated that, in terms of concealing capacity, security and resilience, and imperceptibility, the suggested strategy performs better than other deep learning picture steganography algorithms. The recommended method also benefits from being invisible, and it can produce stego images that are quite similar to the input cover image.

The authors try to fit a full-size color image inside another same size image in [17]. Deep neural networks specially built to cooperate as a pair and are concurrently trained to create the hiding and revealing processes. The system is tested on real-world images of various sources and was trained on images chosen at random from the ImageNet database. The suggested approach compresses and distributes the representation of the secret image across all of the available bits, in contrast to many popular steganographic techniques that encrypt the secret message within the carrier image's least significant bits.

Image steganography techniques are to improve security by embedding data in the carrier image's edge pixels. The suggested method in [18] uses various edge detection filters, such as Prewitt, Sobel, Laplacian, and Canny, to exploit edge-based data concealing in the DCT domain. The PSNR and SNR of the suggested technique's performance are excellent at 99% and 96%, respectively. The message is concealed using edge detection filters. The entire process for an image's red, blue, and green channels occurs in the DCT domain. The proposed steganography method also reduces the size of the images, primarily due to picture compression.

The most crucial information in [19] is that an embedding technique on fusing multiple image features to address any issue developed. This method uses a number of image attributes along the standard characteristic parameters of gray level co-occurrence matrix, image entropy, and form parameter to represent the complexity of the images. The images in the steganographic system, the number and size of images allotted to the stenographer, as other considerations are considered when calculating steganographic capacity. The sub-payload for each chosen image is equal to its estimated capacity. The proposed technique can provide superior security performance against the blind universal pooled steganalysis.

This paper [20] proposes a unique sliding-block segmentation and adaptive steganography-based reversible data hiding (RDH) in an encrypted domain scheme for coding channels. The unique encryption method minimizes the key stream bits while weakening the correlation between neighboring pixels. Experimental research demonstrates that the image can

attain a peak signal-to-noise ratio of >50 dB when using the steganalyser. When the testing error rate is greater than 0.25 with the payload being 0.5 bpp,The developed solution performs better in terms of security than earlier works.

CHAPTER 3

HARDWARE AND SOFTWARE REQUIREMENTS

3.1 Hardware Requirements

- Operating System: Windows 10, 64-bit
- RAM: 8 GB

3.2 Software Requirements

- Platform: Google Colab,VS code
- Programming language: Python
- Libraries: Pandas, Numpy, PIL, OpenCV, etc.,

CHAPTER 4

DATASET DETAILS

The BSDS500, NYUDv2, and PASCAL VOC 2012 datasets were used to train the HED (Holistically-Nested Edge Detection) model for this project. The model can learn to recognise edges in a variety of scenarios.

The Berkeley Segmentation Dataset (BSDS500) is a popular dataset in the field of computer vision for testing edge detection techniques. There are 500 natural photos in total, of which 200 are used for testing, 100 for validation, and 200 for training. The dataset contains excellent edge annotations made by human annotators, offering a trustworthy baseline for assessing edge detection ability. The scenes and objects depicted in the photographs in BSDS500 span a wide spectrum, and the edges seen in the images vary in intricacy and amount of detail.

Using a Kinect camera, the NYU Depth v2 dataset consists of RGB-D images that individually contain depth and color information for each pixel. 1449 indoor scenes with a variety of objects and settings are included in the collection. Similar to the BSDS500 dataset, the edges in the NYU Depth v2 dataset are annotated by humans who place markers at the locations of the edges in the photos.

The PASCAL VOC 2012 dataset serves as a reference set for object detection and segmentation tasks. There are 20 different object types represented by natural photographs with annotations, including people, animals, automobiles, and more. Bounding boxes for objects and segmentation masks are both included in the annotations. The HED model's object detection and edge detection components were integrated after the dataset was used to train the object detection portion of the model.

Overall, deep neural network architecture design, large-scale image dataset curation, and iterative optimization techniques were used to train the HED model. By doing so, the model was able to develop very accurate edge detection skills and successfully generalize to new images.
And then we have collected images from different sources in various categories and each of different sizes to test the model we have implemented in this project.

CHAPTER 5

PROPOSED MODEL ARCHITECTURE

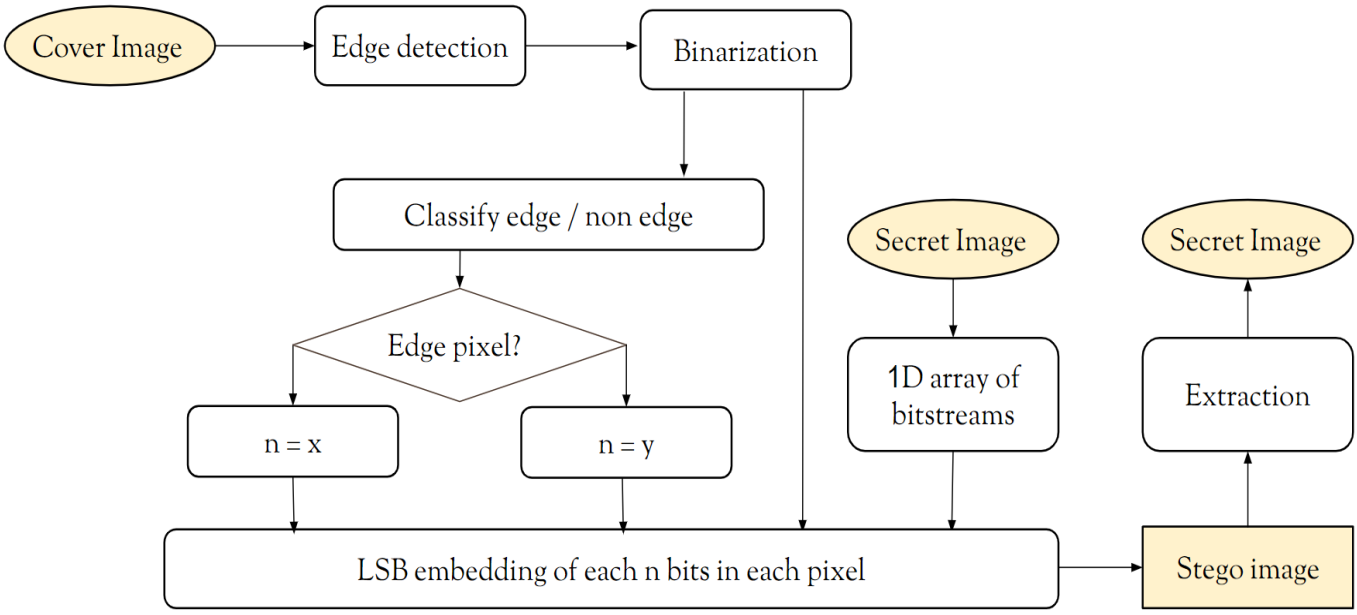


Fig 1. Proposed model architecture

The overall framework of the system is represented by Fig 1. Initially, the sender sends the cover message to the edge detection algorithm which produces an edge detected image as output. Now this edge detected image and secret image undergoes the embedding process where the secret image is converted to 1 dimensional bitstream of 0's and 1's and the binarized image is checked to find the edge pixel if it is an edge, then 'x' number of secret image bits are embedded in cover image (n=x) and if it a non-edge, then 'y' number of secret image bits are embedded in cover image (n=y). And as an output we receive the steganographic image which contains the secret image hidden inside it. Finally this steganographic image is shared with the receiver. The receiver extracts the secret image back from the steganographic image without any leakage of data.

CHAPTER 6

IMPLEMENTATION

6.1 Initial Implementation

In this project, we have a secret image which has the secret message to hide and this image is embedded into an image called a cover image. The final output is called a stego image which can be transmitted into the network. Before embedding the secret image to cover image, we are implementing the edge detecting algorithm so that we will be able to hide more data in the edges and fewer data in the non edges. Then we embed the secret image into the edge detected cover image and also the secret image is extracted back with minimum loss of data. We have also implemented encryption and decryption for further security purposes.

At first, the relevant libraries, including NumPy, and OpenCV, were first imported in the original implementation of this project. Then, we loaded a HED model which had trained and took off its top layers to make room for feature extraction.Afterwards, we used OpenCV to load the cover and secret images. The cover image underwent preprocessing by being resized to the necessary size and having the pixel values normalized. We retrieved feature maps from the preprocessed cover picture using the modified HED model.

We used edge detection techniques on the cover image to uplift the steganographic image's ability of embedding. Using these methods, we were able to extract the cover image's edges, which we then used to modify the feature maps. To embed the secret image, we modified the feature maps using the extracted edges. This step involved altering the feature maps' pixel values at specific locations based on the edges that were extracted. So, while ensuring that the changes made were invisible to the human eye, we were able to store more information within the image.

Finally, we combined the altered feature maps with the cover image to reconstruct the steganographic image. By comparing the steganographic image's PSNR and SSIM to the original cover image, we assessed the image's performance. These metrics enabled us to evaluate the steganographic image's construction quality and information storage capacity. Overall, we were able to use our initial implementation to develop a deep learning-based strategy for image steganography that used edge detection to enhance the steganographic image's embedding capability.

6.2 Challenges faced

The first challenge involves the process of encryption and steganography. When we encrypt data or embed hidden messages within an image, some level of distortion or noise is introduced to the original data or image. This can make it difficult for our model to recover the original image when attempting to decrypt or extract the hidden message. One way to address this is to use more advanced encryption and steganography techniques that introduce less noise and distortion.

The second challenge mentioned is related to the extraction process. When attempting to extract a hidden message or decode an encrypted image, the process can be complex and require careful attention to detail. If the extraction process is not carefully designed and implemented, it may result in errors or an incomplete recovery of the original data or image.

The third challenge mentioned is specific to working with RGB images, which involve multiple color channels. Processing RGB images requires additional steps and techniques to ensure that the colors are properly represented and maintained throughout the embedding and extraction process.

The fourth challenge mentioned involves the issue of image size. When working with pretrained models or existing algorithms, the input image size may need to be modified to match the size that the model expects. This requires serious planning and preprocessing to make sure the input image is properly formatted and sized to work with the model.

6.3 Proposed model implementation

6.3.1 Pre-processing

The cover image and the secret image are loaded using a suitable library, such as OpenCV, as the initial step in preprocessing. The cover image is scaled to the necessary size after loading. This is significant because the pre-trained Holistically-Nested edge detection (HED) model that we'll use for feature extraction needs input images that have a certain size. The cover image's pixel values are normalized after scaling. The process of normalization involves adjusting the pixel values to fall within a predetermined range, between 0 and 1. This step is required because normalizing the input pixel values improves the performance of neural networks like the HED model. By dividing each pixel value in the cover image by the highest pixel value possible, normalization can be achieved. Each pixel value in the image is divided by 255, for instance, if the cover image's maximum pixel value is 255. Pixel values of between 0 to 1 are the result of this. The preprocessed cover image is then fed into the HED model to extract the feature maps. These feature maps serve as a stand-in for the image's content by representing the activations of the neurons in the final convolutional layer of the HED model. We

can incorporate the hidden image into the cover image as it is invisible to the human eye by altering these feature maps. So, in preprocessing, which entails scaling and normalizing the cover picture so that it can be fed into the HED model for feature extraction, is a crucial step in image steganography.

We pre-processes the image by creating a new numpy array P that is same size like the original image (Fig 2) and has each pixel value modified to only include the 3 most significant bits of each channel. This is done to reduce the amount of information in the image and make it more suitable for steganography. Then a nested for-loop that iterates over every pixel in the input image and extracts the 3 most significant bits (MSBs) of each color channel (red, green, and blue) for that pixel. The extracted MSBs are then stored in the corresponding pixel values of the pre-processed image P. The operation applies a bitwise AND operation between the original red channel value and binary value of corresponding pixel, which has the effect of zeroing out the 5 least significant bits of the value and keeping only the 3 most significant bits. The resulting value is then converted to an integer and stored in corresponding pixel value of pre-processed image (Fig 3).The same process is then repeated for the green and blue color channels.

In edge detection, the goal is to identify the boundaries between objects or regions in an image. By extracting only the MSBs of each color channel, the amount of information in the image is reduced, which can help to emphasize the high-contrast edges and make them more visible. This can be useful for edge detection algorithms that rely on identifying sudden changes in pixel values across adjacent pixels. In steganography, the goal is to hide a secret message within an image without it being detected. By extracting only the MSBs of each color channel, the remaining bits can be used to encode the secret message. Because the MSBs represent the most significant information in each color channel, they can be altered slightly without significantly changing the overall appearance of the image. This makes it possible to hide the secret message in a way that is difficult to detect without specialized tools.

Fig 2. Original cover image

Fig 3. Preprocessed cover image

6.3.2 Edge-Detection

Edge detection is a fundamental image processing technique used to identify and locate sharp discontinuities in an image. These discontinuities or edges usually correspond to object boundaries or significant changes in the image's texture or brightness. In other words, edge detection helps to highlight the regions of image where it is a strong change in pixel intensity values.

Compared to conventional and straightforward methods for edge detection, deep learning-based systems have various advantages, including,

- Automatic feature learning: In deep learning-based approaches, the model automatically learns the important features for edge detection from the input data during the training process. This eliminates the need for hand-crafted features, which is time-consuming and may not be as effective.

- Improved accuracy: Deep learning-based approaches achieve better accuracy in edge detection than classic and simple approaches. The reason for this is deep learning models will learn complex non-linear relationships between the input data and output labels, which may be difficult to capture with traditional methods.

- Flexibility: Deep learning models are flexible and are easily adapted to different types of edge detection tasks. They can be trained on large and diverse datasets, making them suitable for a wide range of applications.

- Robustness: Since deep learning models are more resistant to noise and fluctuations in the input data, they are more accurate in detecting edges in practical situations.

The cutting-edge deep learning edge detection algorithm employed in this project is called **HED (Holistically-Nested Edge Detection)**. The HED (Holistically-Nested Edge Detection) model is trained on the BSDS500 dataset, for evaluating edge detection algorithms. The BSDS500 has 500 natural images and their corresponding manually annotated ground-truth edge maps. Here in this method the final edge map is produced by combining a hierarchy of edge predictions from a multi-scale and multi-level CNN (Convolutional Neural Network). By processing the image at various scales and utilizing skip connections to integrate edge information across many layers, the HED algorithm is made to capture edges of various scales and orientations. The outcome is an edge map that faithfully reproduces the edges in the source image, making it a potent steganographic tool. The preprocessing stage of image steganography employs the HED algorithm for edge detection. The function hed preprocess specifically takes the actual image as input, and then employs the HED technique to create an edge map of the image. The cv2.dnn.readNetFromCaffe method is to load a pre-trained model, which is used for implementing the HED algorithm. The HED model's architecture and weights, which were developed using a sizable image dataset, are read into this function. The HED model extracts features at various dimensions and orientations using many layers of convolutional and pooling processes. The image is preprocessed by turning it into grayscale and leveling its pixel values after the HED model has been loaded. The HED model is then applied to the previously preprocessed image to create an edge map. With white pixels denoting the presence of edges and black pixels denoting their absence, the edge map is a binary image that highlights the edges in the input image. The HED function then returns the edge map as in (fig 7) after resizing to match with input image. This cover image then undergoes a binarization process.

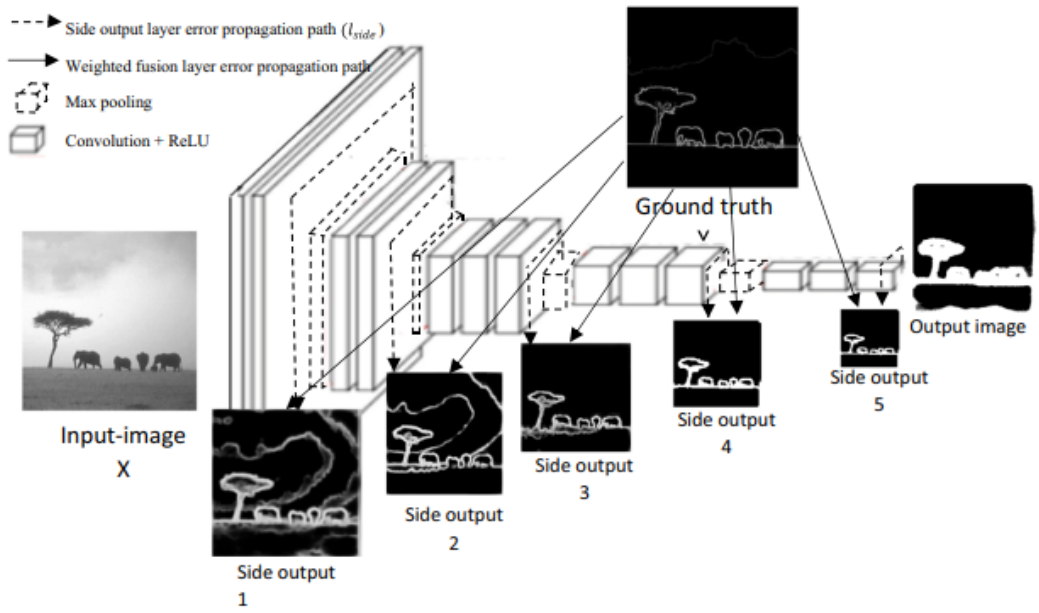


Fig 4. HED implementation

In Fig 4,The error backpropagation paths are highlighted in this diagram of our network architecture for edge detection.Convolutional layers are followed by side-output layers. Each side-output layer is subject to strict control. The side-output-plane size decreases while the receptive field size increases in the HED outputs, which are multi-scale and multilayer. A wide variety of error propagation pathways are used to train the entire network.

A deep learning model called HED performs image-to-image prediction using fully convolutional neural networks and deeply supervised nets. In order to resolve ambiguity in edge and object boundary recognition, HED automatically creates extensive hierarchical representations which are guided by deep supervision on side responses.

MODEL ARCHITECTURE:

With its high density (stride-1 convolutional kernels), considerable depth (16 convolutional layers), and numerous stages (five 2-stride downsampling layers), VGGNet has been shown to attain state-of-the-art performance in the ImageNet challenge.

The HED model is essentially VGGNet with a few changes.- The final convolutional layer in each stage, conv1_2, conv2_2, conv3_3, conv4_3, and conv5, is connected to the side output layer. Each of these convolutional layers has a receptive field size that is the same as the associated side-output layer.The fifth pooling layer and all of the fully connected layers are deleted from the final stage of the VGGNet.The final HED network architecture consists of five stages that are nested inside the VGGNet and have receptive field sizes of 1, 2, 4, 8, and 16 accordingly.

TRAINING HED:

The input size of the network is defined as follows:

- The input layer is named "data".
- The input is a 4D tensor with dimensions of 1 x 3 x 1500 x 1500.
- The first dimension is the batch size (set to 1).
- The second dimension is the number of channels in the image, which is 3 for RGB images.
- The third and fourth dimensions are the height and width of the image, respectively.

The output size of the first layer (conv1_1) as in figure 5 can be calculated as follows:

- The number of filters (num_output) is set to 64 in the convolution parameters.
- The padding (pad) is set to 35, which means that the input is zero-padded with 35 pixels on all sides.

- The kernel size (kernel_size) is set to 3, which means that a 3x3 filter is applied to the input.
- The stride (stride) is not explicitly specified, so it is set to 1 by default.
- The formula for calculating the output size (W2) of a convolutional layer with input size (W1), filter size (F), padding (P), and stride (S) can be expressed as follows:
- W2 = (W1 - F + 2P) / S + 1
- Applying this formula to the first layer, we get:
- W2 = (1500 - 3 + 2*35) / 1 + 1 = 1467
- Therefore, the output of the conv1_1 layer has a size of 64 x 1467 x 1467.

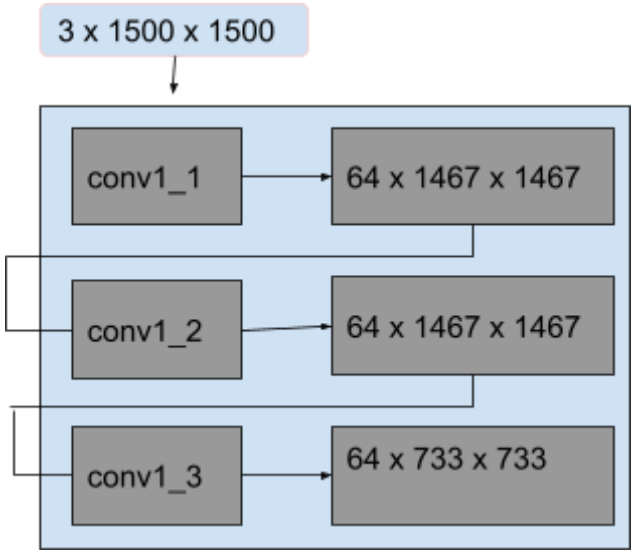


Figure 5: Model diagram of first convolutional layer

The output size of the second layer (conv1_2) is the same as the output size of the first layer (64 x 1467 x 1467) because the convolution parameters and kernel size are the same.

The output size of the pooling layer (pool1) can be calculated as follows:

- The pooling type (pool) is set to MAX.
- The kernel size (kernel_size) is set to 2, which means that a 2x2 filter is applied to the input.
- The stride (stride) is set to 2, which means that the filter is shifted by 2 pixels at a time.
- The formula for calculating the output size (W2) of a pooling layer with input size (W1), kernel size (K), and stride (S) can be expressed as follows:

- W2 = (W1 - K) / S + 1
- Applying this formula to the pooling layer, we get:
- W2 = (1467 - 2) / 2 + 1 = 733
- Therefore, the output of the pool1 layer has a size of 64 x 733 x 733.

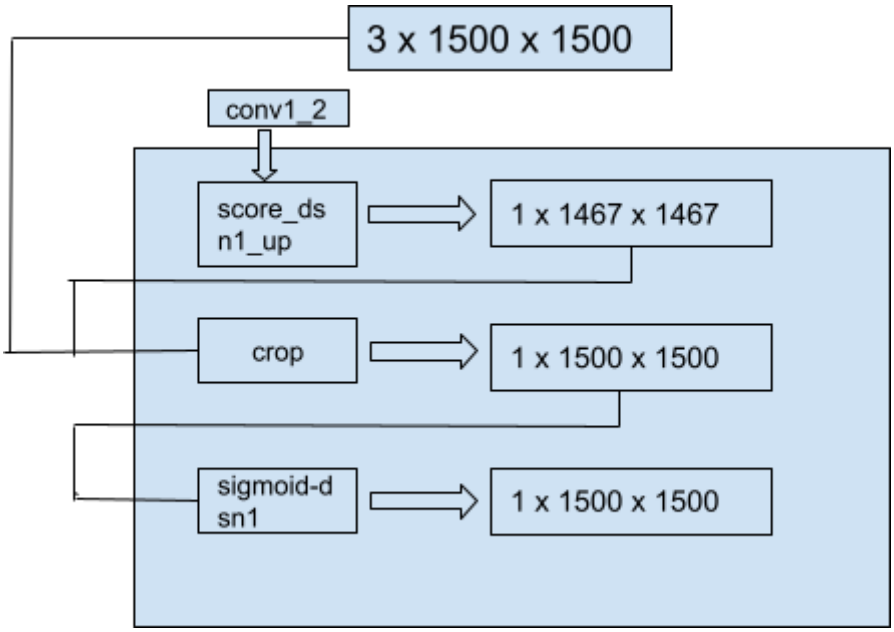


Figure 6. Model diagram of Deep Supervision Network

There are five convolutional networks. First network is described above. Next four convolutional networks will be the same as the first one. But in the third and fourth convolutional network, there will be three convolutional layers instead of two. The output of all five convolutional layers is given as input to each deep supervision network(fig 6) in which upsampling takes place.This is followed by a crop layer in which the output is cropped to the same size as the input image. The sigmoid layer then applies the sigmoid function to the output of the multiscale weight layer to obtain a final binary segmentation map. The concatenation layer combines the output blobs of all five DSN conv layers along the channel dimension to form a single tensor of size 5 x H x W, where H and W are the height and width of the input image, respectively. The multiscale weight layer then applies a 1x1 convolution to the concatenated tensor to obtain a single binary segmentation map of the same size as the input image.

STEPS IN HED:

1. Pre-processing: The input image is pre-processed at this stage to improve its quality and get rid of any noise. Operations like smoothing, scaling, and colour space conversion are frequently used in this step.
2. Multiscale Feature Extraction: In this stage, many scales are applied to the pre-processed image in order to extract features of various sizes. To create a feature map, the image is convolved with a series of filters set at different scales.
3. Multi-level feature fusion: A collection of integrated feature maps is created by fusing the feature maps obtained in the preceding stage at various levels. A collection of fusion modules that merge the feature maps in a hierarchical fashion are used to do this.
4. Side-output prediction: In this step, a collection of side-output maps that represent the edge probabilities at various scales are created using the integrated feature maps.
5. The side-output maps are combined to create a final edge map that represents the edges in the input image during the final edge detection stage

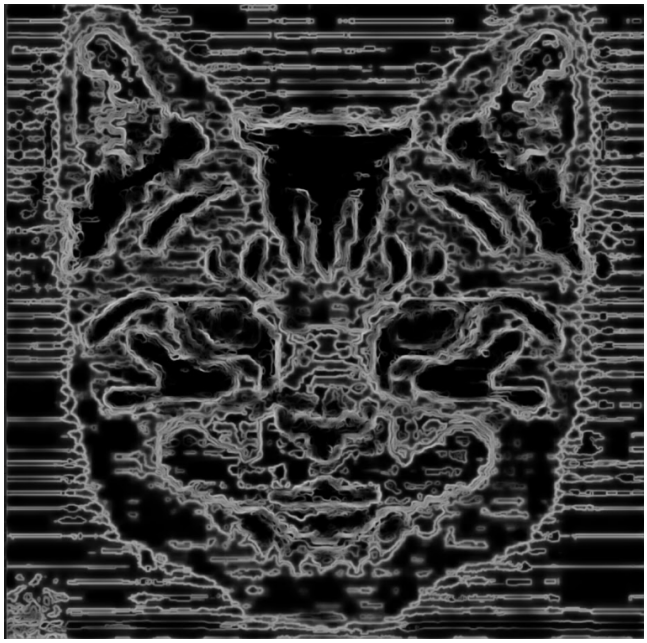


Fig 7. Cover image after HED Edge detection.

Another edge detection algorithm employed in this project is **The Sobel edge detection method**. This is implemented using convolutional filters. The method involves convolving the input image with two separate filters in the x and y direction, respectively. These filters are called Sobel kernels or Sobel operators. To perform the edge detection, the input image is first convolved with the Sobel kernel in the x direction, resulting in an output image that highlights edges that are vertical or slanted to the right. Similarly, the input image is convolved with the Sobel kernel in the y direction, resulting in an output image that highlights edges that are horizontal or slanted to the left. The magnitude of the edge is then calculated as the square root of the sum of the squared gradients in the x and y directions, respectively. This edge map is a

grayscale image that highlights the edges in the original input image. Finally this Sobel function then returns the edge map as in (fig 8).

Fig 8. Cover image after Sobel Edge detection.

Sobel is a classical edge detection method that is based on convolution with specific filter kernels (Sobel operators) in the vertical and horizontal directions. It calculates the gradient magnitude of an image,, which expresses how quickly the image intensity changes at each pixel. This method works good for simple images with distinct edges, but it may not perform well on complex images with noisy or textured backgrounds(fig 6). HED, on the other side, is a more recent and advanced process for edge detection that uses deep neural networks. It is the idea of holistically-nested edge detection, which is that the model is designed to simultaneously detect edges at multiple scales and levels of abstraction. HED uses a deep convolutional neural network trained on a large dataset of annotated images to predict edge maps for new images. This method performs good on more images with varying complexities, and had shown to outperform other edge detections, including the Sobel operator.

6.3.3 Post- Processing

Binarization is often applied to edge detected outputs to receive a binary image where the edges are clearly visible and easy for more processing, such as object recognition or segmentation. By thresholding the grayscale edge detection output and setting a threshold value, pixels above the threshold are set to white and pixels below the threshold are set to black. This produces a binary image as its edges are represented as white lines on a black background, making it easy for extracting meaningful information about the edges in the image.

The binarization takes place during the post-processing step, after the modified feature maps have been obtained. Binarization comes next after the edge detection and embedding phase. By thresholding the pixel values, binarization transforms a grayscale image into a binary image. To make the embedded image easily concealable in the cover image, binarization is performed in your code to turn it into a binary image. The Python OpenCV package is used to do the binarization procedure. The cv2.threshold() function, which accepts the embedded picture and a threshold value as arguments, is specifically employed. All the pixels in the embedded picture with values more than the threshold value are set to 255 (white), and all the pixels with values equal to or less than the threshold value are set to 0. (black).

The midpoint of the grayscale pixel values, or 127, is the threshold value (ranging from 0 to 255). The binary image has a good balance of black and white pixels, which is necessary for the successful concealing of the embedded image in the cover image, hence this threshold value were set to achieve that. The image is preprocessed by turning it into grayscale and leveling its pixel values after the HED model has been loaded. The HED model is then applied to the previously preprocessed image to create an edge map. With white pixels denoting the existence of edges and black pixels denoting their absence, the edge map is a binary image that highlights the edges in the input image. The HED postprocess function then returns the edge map after resizing. The secret picture is then embedded in the cover image using this edge map.

And the image is subjected to the following four thresholding methods:

- Otsu thresholding: A global thresholding technique that automatically calculates an optimal threshold value based on the histogram of the image. The threshold value is calculated to minimize the intra-class variance of the foreground and background pixels. The output image of Otsu thresholding is displayed in Fig 9.

- Hard thresholding: It is a simple thresholding technique where a fixed threshold value is applied to the image to binarize it. In Fig 10,Hard thresholding image is shown.

- Mean adaptive thresholding: This is a local thresholding technique where calculating threshold value for each pixel by considering the mean of its neighboring pixels. The block size and constant value parameters can be adjusted. In Fig 11, the output of the Mean adaptive thresholding image is shown.

- Gaussian adaptive thresholding: This is another local thresholding technique where the threshold value is calculated for each pixel based on the weighted sum of its neighboring pixels using a Gaussian window. The block size and constant value parameters can be adjusted to control the size of the neighborhood and the threshold value.Fig 12,shows the output of the Gaussian adaptive thresholding image.

Fig 9. Otsu thresholding

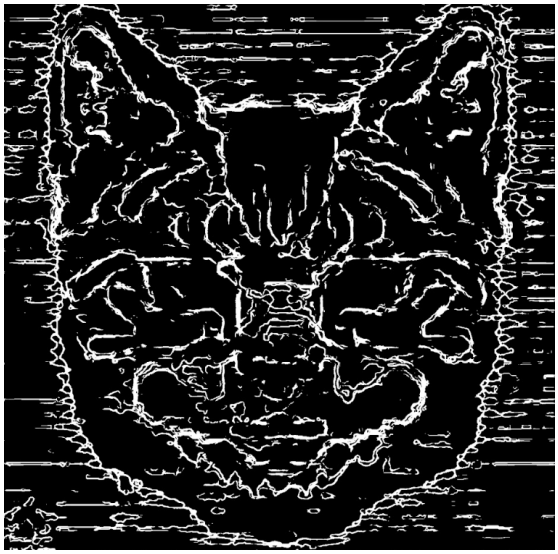


Fig 10. Hard thresholding

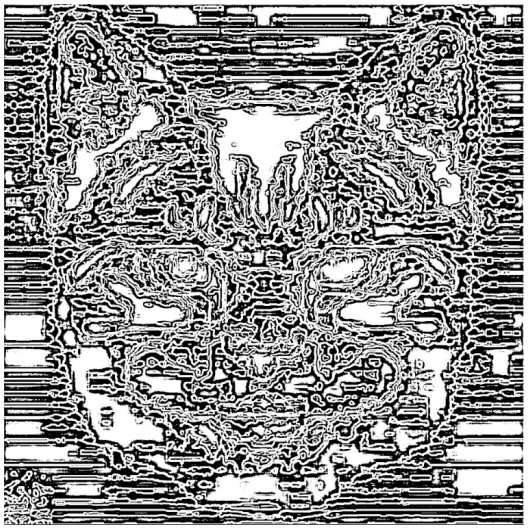


Fig 11. Mean adaptive thresholding

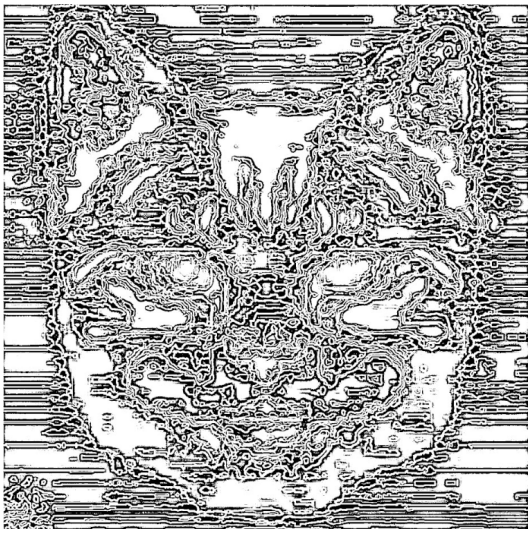


Fig 12. Gaussian adaptive thresholding

Now comparing among all four thresholding techniques, The Gaussian adaptive thresholding technique provides the better result as this method gives the maximum number of edges. So, we are proceeding this project with Gaussian adaptive thresholding technique.

MODELS	Otsu thresholding	Hard thresholding	Mean adaptive thresholding	Gaussian adaptive thresholding
NO.OF.EDGES				
HED	660635	185788	807123	**952832**
SOBEL	188952	17420	614313	780863
CANNY	74259	74259	899352	926658
VGG16	461273	0	79935	20506

Table 1. Comparing the number of edges of four models

Table 1 compares the four binarization methods that are used following the preprocessing step.The majority of the four binarization methods exhibit a greater number of edges when using Gaussian adaptive thresholding.HED produces more edges than the other four implemented models.

6.3.4 Embedding and Extraction the secret image

From Fig 1, we can understand the overall architecture of our project which includes embedding and extraction of the secret image. The embedding process starts after the binarization process. Using gaussian thresholding technique, the edge map derived from the HED model is transformed into a binary picture in the above section.

Steps for Embedding the secret image:

1. Read the cover image, secret image, and a thresholded version of the cover image.
2. Convert the secret image into a 1D bitstream.
3. The number of bits embedded in each pixel depends on whether it is an edge pixel or a non-edge pixel. For edge pixels, x bits are embedded, and for non-edge pixels, y bits are embedded
4. Generate a random index and convert them into binary and store them in the channel R of RGB.
5. Check whether the pixel is edge or non edge, and store the particular random number of secret bits in channel G and B of RGB.
6. After embedding all the bits of the secret image, the steganographic image is saved.

The output file is then stored with the steganographic image. In general, the embedding technique entails concealing the binary message within the cover image's edge map.

The embedding process is reversed during the extraction process.The key used to encrypt the image during embedding is the same key used for steganography. The steganographic key and the steganographic picture—the image containing the concealed image—are inputs used to start the extraction process.

Steps for Extracting the secret image:

1. Read the cover image, steganographic image, thresholded version of the cover image.
2. The number of bits extracted in each pixel depends on if it is an edge pixel or a non-edge pixel. For edge pixels, x bits are extracted, and for non-edge pixels, y bits are extracted.
3. Retrieve the last two bits of channel R of RGB and convert them into decimal.
4. Check whether the pixel is edge or non edge, and retrieve the particular decimal number of secret bits from channel G and B of RGB.
5. Pack all the secret bits and convert them into a numpy array.
6. After extracting all the bits of the secret image, the secret image is saved.

Now successfully the secret image is extracted back from the cover image. In (fig 13) the overall architecture of the project with images of every stage is explained clearly.

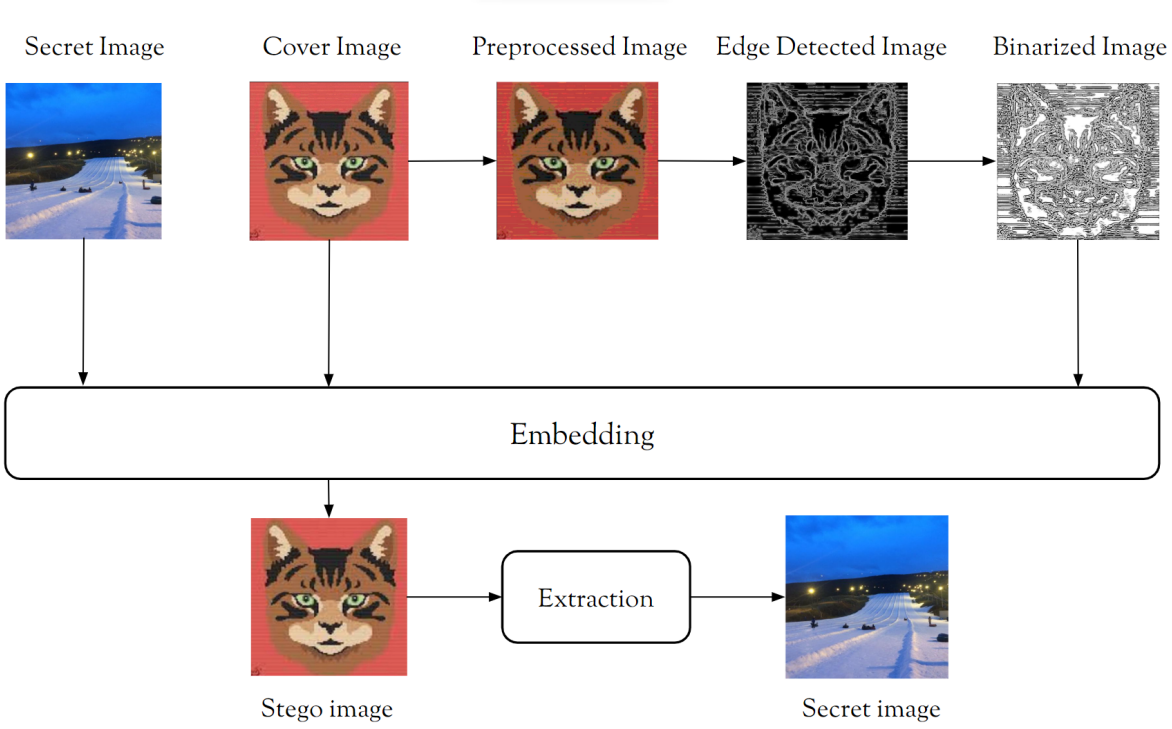


Fig 13. Proposed model output

CHAPTER 7

PERFORMANCE METRICS

7.1 Metrics

7.1.1 Peak Signal - Noise Ratio (PSNR)

Peak signal-to-noise ratio (PSNR) is a term for the relationship between the maximum allowed value (power) of a signal and the maximum power of distorted noise that reduces the signal's ability to be accurately represented. Because many signals have a very wide dynamic range, the PSNR is often represented in terms of the logarithmic decibel scale (ratio between the largest and smallest possible values of a changeable quantity). The image quality is better and there is less distortion the higher the PSNR number.

```
MSE = np.mean((CI - SI_prime)**2)
PSNR = 10 * np.log10(255 / np.sqrt(MSE))
```

Where,CI - cover image, SI_prime - stego image, MSE- Mean square error

7.1.2 Payload

The term "payload capacity" refers to the largest message size that can be encoded in a cover image, and it is commonly expressed in bits per pixel. (bpp). However, the amount that the cover image's pixel value changes will depend on the size of the message that is pinned to it.

```
Payload = (1- N/(R*C))*y + (N/(R*C)) * x
```

Where R,C - height and width of the image, N - number of edge pixels, x,y - number of bits to embed

7.1.3 Structural Similarity Index (SSIM)

A measure that determines how similar two images are to one another is called the Structural Similarity Index (SSIM). The SSIM value ranges from -1 to 1, where -1 denotes a complete difference between the two images and 1 denotes an exact match. The common consensus is that a score of 0.9 or above indicates a high degree of resemblance, whilst values

between 0.7 and 0.9 indicate a moderate degree of similarity and values below 0.7 indicate a low degree of similarity.

$$l = (2 * mean1*mean2 + C1)/(mean1**2 + mean2**2 + C1)$$

$$c = (2*std1*std2 + C2)/(std1**2 + std2**2 + C2)$$

$$SSIM = (covariance + C3)/(std1*std2 + C3)$$

Where C1, C2, C3 - constants

7.1.4 Histogram similarity score

Histogram similarity score is a measure that is used to determine the similarity between two histograms. This is used to compare the distributions of pixel values in different images or to compare the distribution of features in different images.The correlation coefficient is used as the histogram similarity score, which is a common method for comparing histograms.

$$\text{Histogram similarity score} = (corr\_r + corr\_g + corr\_b) / 3$$

where corr_r, corr_g, and corr_b - correlation coefficients for the red, green, and blue channels of the images

7.1.5 Chi square distance

A statistical technique called chi-square distance calculation used for comparing two feature matrices.By comparing the distribution of pixel intensities in the original image to that of the steganographic image, the χ^2 distance can be used to detect the presence of hidden data.

The Chi-square distance of 2 arrays ‘x’ and ‘y’ with ‘n’ dimension is mathematically calculated using below formula :

$$d(x,y) = sum((xi-yi)^2 / (xi+yi)) / 2$$

	PSNR	Payload	Similarity Index	Histogram similarity score	Chi-square
HED	41.85	1.42	0.9976	0.97	0.03
Sobel	38.04	1.58	0.9279	0.95	0.03
Canny	38.07	1.41	0.9573	0.94	0.03
VGG16	37.04	1.21	0.9361	0.93	0.02

Table 2. Performance Metrics Comparison

The performance metrics of the various edge detection algorithms are compared in Table2. The comparison showed that HED could detect edges more precisely than other edge detection methods. The implemented edge detection methods that have been used here,uses the same embedding and extraction approach. HED has values higher than the other methods because, in contrast to all other edge detection techniques, it finds the right edges.

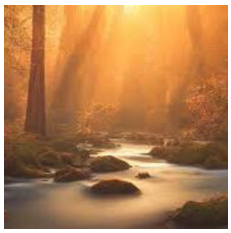
CHAPTER 8

CONCLUSION

In this research, we looked into the concept of image steganography, a technique for hiding private data in pictures. The proposed method makes use of edge detection and binary image conversion techniques to embed one image inside another. It is a fact that the human eye can distinguish a clear difference between two image pixels. This shows that the borders of a picture can withstand more distortion than the other areas of the cover image since they alone have a sharp shift in pixel values from the surrounding areas. In this work, we applied an edge-based picture steganography technique to hide one image behind another.

To sum up, HED is a more sophisticated edge detection technique that uses deep neural networks to train features for edge detection, in contrast to Sobel, a traditional edge detection technique that depends on hand-crafted filter kernels. While Sobel is more computationally efficient and easier to implement, HED is more reliable and accurate at spotting edges in complex images where other edge detection produces false edges.

To get a higher payload with the least amount of distortion in the steganographic images, we have presented a HED edge detection based steganography approach. In this instance, a HED edge detection is used to compute the edges of the cover images. Our results, which show that four distinct binarization procedures provide a virtually same result in terms of data hiding capabilities, demonstrate that the suggested edge detection strategy is less sensitive to the thresholding approaches used during binarization and also produces a higher number of correct edges. While maintaining a respectable payload capacity of (3.67) for a single channel, the higher PSNR(41.85) and SSIM of (0.9976) obtained by the suggested method beats several existing techniques in both the spatial and edge-based Steganography methods.

Environmental condition	HED	SOBEL	CANNY	VGG16
	No.of.edge pixels: 1027017	No.of.edge pixels: 1255729	No.of.edge pixels: 1769236	No.of.edge pixels: 285147
	No.of.edge pixels: 863441	No.of.edge pixels: 1234824	No.of.edge pixels: 1854241	No.of.edge pixels: 458405
	No.of.edge pixels: 954354	No.of.edge pixels: 931632	No.of.edge pixels: 1293915	No.of.edge pixels: 1067399
	No.of.edge pixels: 1448339	No.of.edge pixels: 1699885	No.of.edge pixels: 2160704	No.of.edge pixels: 1872582

	**No.of.edge pixels: 1080161**	**No.of.edge pixels: 1288783**	**No.of.edge pixels: 1483065**	**No.of.edge pixels: 528263**

Table 3. No.of.edges pixel for all four methods

The Sobel and Canny approaches produce more edges than HED due to their simpler and more direct approach, according to Table 3's comparison of the edges produced utilizing the four different methods. In contrast, HED is a more sophisticated and computationally demanding method that aims to generate fewer but more accurate edges.HED produces fewer edges, but its PSNR value is higher than that of other techniques that generate more edges from Table 2.

SECRET IMAGE SIZE	PSNR	PAYLOAD	SSIM	HISTOGRAM SCORE	CHI-SQUARE DISTANCE
100 x 100	47.27	1.42	0.9998	0.99	0.00
200 x 200	46.00	1.42	0.9995	1.00	0.00
300 x 300	42.46	1.42	0.9991	0.99	0.01
400 x 400	39.75	1.42	0.9998	0.99	0.01
500 x 500	41.85	1.42	0.9976	0.97	0.03

Table 4. Performance metric using different secret image size

Table 4 shows the maximum size of a secret image that can be embedded in the cover image.Five different image size combinations have been exhausted, and their performance metrics only slightly differ from one another.

BIBLIOGRAPHY

[1] Ray, B., Mukhopadhyay, S., Hossain, S., Ghosal, S. K., & Sarkar, R. (2021). Image steganography using deep learning based edge detection. Multimedia Tools and Applications. doi:10.1007/s11042-021-11177-4

[2] S. Xie and Z. Tu, "Holistically-Nested Edge Detection," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1395-1403, doi: 10.1109/ICCV.2015.164.

[3] Shashikala Channalli, Ajay Jadhav, “Steganography An Art of Hiding Data”,International Journal on Computer Science and Engineering Vol.1(3), 2009, 137-141

[4] Srilekha Mukherjeea, Subhajit Roya , Goutam Sanyal,”Image Steganography Using Mid Position Value Technique”,International Conference on Computational Intelligence and Data Science,132 (2018) 461–468

[5] S. Pramanik, D. Samanta, S. Dutta, R. Ghosh, M. Ghonge and D. Pandey, "Steganography using Improved LSB Approach and Asymmetric Cryptography," 2020 IEEE International Conference on Advent Trends in Multidisciplinary Research and Innovation (ICATMRI), 2020, pp. 1-5, doi: 10.1109/ICATMRI51801.2020.9398408.

[6] N. M. Al-Aidroos and H. A. Bahamish, "Image Steganography Based on LSB Matching and Image Enlargement," 2019 First International Conference of Intelligent Computing and Engineering (ICOICE), 2019, pp. 1-6, doi: 10.1109/ICOICE48418.2019.9035172.

[7] A. Pabbi, R. Malhotra and K. Manikandan, "Implementation of Least Significant Bit Image Steganography with Advanced Encryption Standard," 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), 2021, pp. 363- 366, doi: 10.1109/ESCI50559.2021.9396884.

[8] R. R. Isnanto, R. Septiana and A. F. Hastawan, "Robustness of Steganography Image Method Using Dynamic Management Position of Least Significant Bit (LSB)," 2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), 2018, pp. 131-135, doi: 10.1109/ISRITI.2018.8864439.

[9] Mohammed A. Saleh, Image Steganography Techniques, International Journal of Advanced Research in Computer and Communication Engineering Vol. 7, Issue 9, September 2018, page no 52 - 58, DOI 10.17148/IJARCCE.2018.7910

[10] Abhishek Das, Japsimar Singh Wahi, Mansi Anand, Yugant Rana, Multi-Image Steganography Using Deep Neural Networks, International Journal of Electrical and Computer Engineering (IJECE), 2 Jan 2021.

[11] Rusul Mohammed Neamah, Jinan Ali Abed, Elaf Ali Abbood, Hide text depending on the three channels of pixels in color images using the modified LSB algorithm, International Journal of Electrical and Computer Engineering (IJECE), Vol. 10, No. 1, February 2020, pp. 809~815 ISSN: 2088-8708, DOI: 10.11591/ijece.v10i1.pp809-815

[12] Huda Kadhim Tayyeh, Ahmed Sabah Ahmed Al-Jumaili, A combination of least significant bit and deflate compression for image steganography, International Journal of Electrical and Computer Engineering (IJECE), Vol. 12, No. 1, February 2022, pp. 358~364 ISSN: 2088-8708, DOI: 10.11591/ijece.v12i1.pp358-364

[13] Nashat, D., Mamdouh, L. “An efficient steganographic technique for hiding data”. J Egypt Math Soc 27, 57 (2019). https://doi.org/10.1186/s42787-019-0061-6

[14] Manikandan, Vazhora Malayil, and Vedhanayagam Masilamani. "An improved reversible data hiding scheme through novel encryption." 2019 Conference on Next Generation Computing Applications (NextComp). IEEE, 2019.

[15] Elharrouss, Omar, Noor Almaadeed, and Somaya Al-Maadeed. "An image steganography approach based on k-least significant bits (kLSB)." 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT). IEEE, 2020.

[16] Nandhini Subramanian; Ismahane Cheheb; Omar Elharrouss; Somaya Al-Maadeed; Ahmed Bouridane,”End-to-End Image Steganography Using Deep Convolutional Autoencoders” IEEE Access (Volume: 9),20 September 2021,DOI: 10.1109/ACCESS.2021.3113953.

[17] Shumeet Baluja,”Hiding Images in Plain Sight: Deep Steganography”,31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

[18] Ayub, N., & Selwal, A. (2020). An improved image steganography technique using edge based data hiding in the DCT domain. Journal of Interdisciplinary Mathematics, 23(2), 357– 366. doi:10.1080/09720502.2020.1731949

[19] Yang, J., & Liao, X. (2020). An Embedding Strategy on Fusing Multiple Image Features for Data Hiding in Multiple Images. Journal of Visual Communication and Image Representation, 102822. doi:10.1016/j.jvcir.2020.102822

[20] Kun Liang Yu, Liquan Chen, Yu Wang, Jinguang Han, Lejun Zhang. Reversible data hiding in encrypted images for coding channel based on adaptive steganography doi:10.1049/iet-ipr.2020.1105

[21] N. Subramanian, O. Elharrouss, S. Al-Maadeed and A. Bouridane, "Image Steganography: A Review of the Recent Advances," in IEEE Access, vol. 9, pp. 23409-23423, 2021, doi: 10.1109/ACCESS.2021.3053998.

[22] SongulKarakus, EnginAvci, “A new image steganography method with optimum pixel similarity for data hiding in medical images”,June 2020, https://doi.org/10.1016/j.mehy.2020.109691

[23] Pramanik, S., Bandyopadhyay, S. K., & Ghosh, R. (2020). Signature Image Hiding in Color Image using Steganography and Cryptography based on Digital Signature Concepts. 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA). doi:10.1109/icimia48430.2020.9074957

[24] Hussain H. AlyasAlharith,A. Abdullah Alharith,A. Abdullah,”Enhancement the ChaCha20 Encryption Algorithm Based on Chaotic Maps”,January 2021,DOI: 10.1007/978-981-16-0666-3_10.

[25] Kartik Sharma1, a, * , Ashutosh Aggarwal1, b, Tanay Singhania2, c, Deepak Gupta1, d, Ashish Khanna1, e,”Hiding Data in Images Using Cryptography and Deep Neural Network”,Journal of Artificial Intelligence and Systems, 2019, 1, 143-162,ISSN Online: 2642-2859