

# Enhanced CNN Approaches for Multi-Image Embedding in Image Steganography

by

Md. Irtiza Hossain

20101481

Samiul Kadir

20101211

Farhan Ishraq Fagun

20101295

Ishtiaq Samiul

20101133

Rafi Zaman Saukhin

20301143

A thesis submitted to the Department of Computer Science and Engineering  
in partial fulfillment of the requirements for the degree of  
B.Sc. in Computer Science

Department of Computer Science and Engineering  
Brac University  
May 2024


© 2024. Brac University  
All rights reserved.

# Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

## Student's Full Name & Signature:



---

Md. Irtiza Hossain  
20101481



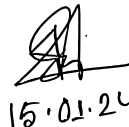
---

Farhan Ishraq Fagun  
20101295



---

Samiul Kadir  
20101211



15.01.24

---

Ishtiaq Samiul  
20101133



---

Rafi Zaman Saukhin  
20301143

# Approval

The thesis/project titled "Enhanced CNN Approaches for Multi-Image Embedding in Image Steganography" submitted by

1. Md. Irtiza Hossain (20101481)
2. Samiul Kadir (20101211)
3. Farhan Ishraq Fagun (20101295)
4. Ishtiaq Samiul (20101133)
5. Rafi Zaman Saukhin (20301143)

Of Spring, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May, 2024.

## Examining Committee:

Supervisor:  
(Member)



---

Dr. Muhammad Iqbal Hossain  
Associate Professor  
Department of Computer Science and Engineering  
Brac University

Program Coordinator:  
(Member)

---

Dr. Md. Golam Rabiul Alam  
Professor  
Department of Computer Science and Engineering  
Brac University

Head of Department:  
(Chair)

---

Dr. Sadia Hamid Kazi  
Chairperson and Associate Professor  
Department of Computer Science and Engineering  
Brac University

# Abstract

In today's world of information and communication tools, data security is critical for information diffusion. With the growth of extensive multimedia sharing and secret discussions, data concealment has become increasingly vital. Steganography encompasses various types, including image steganography, audio steganography, video steganography, text steganography, network steganography, and digital watermarking. Traditionally, image steganography involves concealing an image within the least significant pixels of a cover image. However, recent advancements have leveraged neural networks to encode and decode secret images within cover images. Our objective is to utilize neural networks especially convolutional neural network to hide multiple images within a single cover image while maximizing payload capacity and minimizing errors in the encoding and decoding processes.

**Keywords:** Steganography; Image Steganography; Neural Network; Convolutional Neural Network

## **Acknowledgement**

Firstly, all praise to the Great Allah for whom our thesis has been completed without any major interruption.

Secondly, to our supervisor Dr. Muhammad Iqbal Hossain sir for his kind support and advice in our work. He helped us whenever we needed help.

And finally to our parents without their support, it may not be possible. With their kind support and prayer, we are now on the verge of our graduation.

# Table of Contents

Declaration	i
Approval	ii
Abstract	iii
Acknowledgment	iv
Table of Contents	v
List of Figures	vii
List of Tables	viii
Nomenclature	ix
<b>1 Introduction</b>	<b>1</b>
1.1 Research Problem . . . . .	2
1.2 Research Objectives . . . . .	3
1.3 Why Using <i>CNN</i> . . . . .	3
1.4 Thesis structure . . . . .	4
<b>2 Literature Review</b>	<b>5</b>
2.1 Background . . . . .	5
2.1.1 Image Steganography . . . . .	5
2.1.2 Neural Networks . . . . .	6
2.2 Related Works . . . . .	7
<b>3 Dataset Description</b>	<b>14</b>
3.1 Description of the Data . . . . .	14
3.1.1 ImageNet . . . . .	14
3.1.2 Cifar10 . . . . .	15
<b>4 Methodology</b>	<b>16</b>
4.1 Working Process . . . . .	16
4.2 Description of the Model . . . . .	17
4.2.1 Encoder Model . . . . .	17
4.2.2 Decoder Model . . . . .	21
4.2.3 Full Model . . . . .	24

<b>5</b>	<b>Result Analysis</b>	<b>28</b>
5.1	Two Image Steganography . . . . .	28
5.2	Three Image Steganography . . . . .	34
<b>6</b>	<b>Implementation and Analysis of Security Measures</b>	<b>42</b>
6.1	Implementation of Cryptographic Algorithms in Our Model . . . . .	43
6.2	Architectural Key Integration for Enhanced Security . . . . .	45
<b>7</b>	<b>Conclusion</b>	<b>49</b>
	<b>Bibliography</b>	<b>51</b>

# List of Figures

2.1	Simple Classification of Image Steganography Methods Proposed by [2]	5
2.2	Simple Neural Network Structure [24]	7
4.1	Working Process of Our Full Model for 2 Image Steganography	16
4.2	Working Process of Our Full Model for 3 Image Steganography	17
4.3	Encoder Model for 2 Image Steganography	18
4.4	Encoder Model for Three Image Steganography, Similar to Two Image Steganography	19
4.5	Final Encoder Model for Three Image Steganography	20
4.6	Decoder Model	22
4.7	Full Model	26
4.8	Full Model Workflow	27
5.1	Distribution of Errors in Cover and Secret Images using Lrelu Activation	29
5.2	Result of Concealing Two Secret Images.	30
5.3	Result of Concealing Two Secret Images (Without Noise)	32
5.4	Distribution of Errors in Cover and Secret Images	35
5.5	Result of Concealing Three Secret Images.	36
5.6	Result of Concealing Three Secret Images (Without Gaussian noise in the Decoder).	39
5.7	Result of Concealing Two Secret Images in Plain White Cover Image.	41
5.8	Result of Concealing Two Full Black Secret Image in Plain White Cover Image.	41
6.1	Result of Removing the Last Two Convolutional Layers from Decoder	43
6.2	Workflow of Our Model with Encryption and Decryption	44
6.3	Result of Using AES	44
6.4	Result of Using ChaCha20	45
6.5	Result of Using Architectural Key Integration	46
6.6	Result of Using Architectural Key Integration	47



# List of Tables

5.1	Error Per Pixel . . . . .	28
5.2	Accuracy Per Pixel . . . . .	29
5.3	Performance Metrics of figure 5.2, Result of Concealing Two Secret Images and Comparison with [15] . . . . .	31
5.4	Overall Performance Comparison of Proposed Model With and Without Gaussian Noise . . . . .	32
5.5	Performance Metrics of figure 5.3, Result of Concealing Two Secret Images (Without Noise) . . . . .	33
5.6	Performance Metrics for Concealing Multiple Secret Images . . . . .	34
5.7	Performance Metrics for Concealing Three Secret Images . . . . .	38
5.8	Performance Metrics for Concealing Three Secret Images (Without Gaussian noise in the Decoder) . . . . .	40
6.1	Comparison of AES and ChaCha20 Performance Metrics . . . . .	45
6.2	Architectural Key Integration Performance Metrics . . . . .	46
6.3	Performance Metrics with Incorrect Key in Decoder . . . . .	47
6.4	Comparison of AES and ChaCha20 Performance Metrics . . . . .	47

# Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

*BPP* Bits Per Pixel

*CNN* Convolutional Neural Network

*HVS* Human Visual System

*ISN* Invertible Steganography Network

*LSB* Least Significant Bit

*MSE* Mean Square Error

*PSNR* Peak Signal to Noise Ratio

*SSIM* Similarity Index Measurement

# Chapter 1

## Introduction

Steganography is a captivating field that involves the core concepts of information hiding and cryptography. It offers an array of techniques and computational protocols to veil confidential data, allowing its dissemination or retention without any suspicion. Whereas cryptography mainly focuses on obscuring the substance of a communication, steganography centers on hiding the very existence of a message. Steganography is an effective and important technique for secret communication and protecting data in different fields. The historical significance of this may be traced back to ancient civilizations. This civilization used complicated techniques example wise invisible ink, microdots as well as secret compartments to send classified information. In this modern era of tech advancement, steganography has made huge progress. It applies the vast and flexible information included in digital documents. This has made it a crucial tool for secretly sharing and protecting data. The inability to conceal information in a conspicuous manner without generating suspicion makes steganography an insignificant tool for intelligence agencies, businesses, and individuals aiming to protect their sensitive information.

In steganography, a questionable approach is used where information is concealed by replacing the least significant bit LSB of a pixel in a picture with a bit that contains the hidden data. This technique provides a straightforward approach for transmitting secret data. However, LSB steganography is restrained by its constrained embedding capacity, which curtails the amount of data that can be concealed within the least significant bits of the cover media. Additionally, LSB steganography is susceptible to detection and can be easily compromised through basic statistical analysis or image processing methods. [21].

The security and robustness of steganographic systems can be greatly improved through the use of deep learning, as researchers have increasingly recognized in recent years. Due to the outcome, they have started exploring the implementation of neural networks, especially *CNN*, for the purpose of image steganography. Neural steganography endeavors to heighten the security and efficiency of steganographic systems by exploiting the capabilities of *CNN*'s to learn intricate features and patterns from images. This ability enables the more sophisticated and secure embedding of hidden data while preserving the visual quality of the cover media. Thus, cognitive steganography reveals incredible potential in the domain of data safety and can pave the way for even more sophisticated and secure transmission of sensitive

information [21].

## 1.1 Research Problem

In the current digital era, the transfer of multimedia data has become so rampant that securing and preserving the privacy of shared data has become a crucial issue. As encryption is used to protect data alongside standard steganography methods also protects the data during transmission there is still a chance of flaws as well as the risk of being detected. It is crucial to investigate new methods. Because by new methods the security of multimedia data transport protocol improves. There exists an urgent necessity to enhance the measures pertaining to security. The initial strategy employed in the realm of image steganography is known as the Least Significant Bit, denoted as the *LSB* methodology. However, it possesses a few shortcomings. One of them is the incapability of this particular system to withstand visual incidents and steganalysis algorithms, thereby constituting a significant vulnerability. Additionally, the *LSB* approach demonstrates a limited capacity for embedding, thereby allowing for the concealment of only a minimal quantity of data within an image. In addition, it is vulnerable for the compression of lossy in order to alter or deteriorate the concealed data as it is being compressed. Experts have suggested using advanced steganography techniques. Example wise deep learning . This address has these issues and boost the system's resilience as well as security. But further research is required to develop steganography methods that are both secure and efficient enabling us to overcome the constraints of the *LSB* methodology. In the future, novel methodologies may arise that will allow us to hide extensive quantities of data within photos without being detectable to visual attacks or the steganalysis algorithms. Till then scientists must continue to enhance and refine steganography techniques in order to enhance security and resilience.

In recent works the researcher introduces a novel technique in neural image steganography that overcomes many of the limitations of previous works. This trailblazing endeavor establishes a novel trajectory for steganography, employing the prowess of neural networks to cloak hidden messages within images. Despite its effectiveness, the method does have some limitations, such as its inability to handle images with a lot of text or sharp edges, which are more susceptible to visual artifacts. The deficiency of the approach is its susceptibility to compression, as compressed images might lose some of the concealed data. Additionally, the work fails to tackle the problem of steganalysis, the detection of covert messages within images.

In another scholarly work the researchers delved into the topic of how *CNNs* outperform other architectures like DenseNets, FC-DenseNets, and R-*CNN*. However, the study did not extensively touch on the crucial aspect of security evaluation in the context of *CNN*-based steganography. Security should be a top priority in steganography to guarantee that our confidential data remains hidden within our secret image and is not accessible to unauthorized parties. Therefore, scrutinizing the safety implications of *CNN*-based camouflage tactics is a crucial domain of study that necessitates further scrutiny.

As we are working with multi-image steganography which can be compared with the

work described above. Their main focus was to store a maximum number of secret images in a single cover image with minimal error. While doing this, their image after the encoding was blurry so we can assume there was a bit too much error and they did not talk about the size of the image after the encoding phase.

Our research focuses on making secret images easier to decode accurately, as well as enhancing the quality of the original images used to hide them.

## 1.2 Research Objectives

The primary objective of this research is to formulate an innovative steganography approach by utilizing deep neural networks to allow for the concealment of a maximal number of covert images within a single cover image with minimal errors throughout the encoding and decoding stages. LSB steganography, a traditional technique, has limitations in terms of embedding capacity and vulnerability to detection, which are also true for neural image steganography due to the computational complexity linked with training and utilizing deep neural networks. The preparation of artificial intelligence systems necessitates a substantial quantity of information and computational resources and may be time-consuming. Therefore, the objective of this research is to apply the capabilities of deep neural networks, specifically *CNN*, to conquer these restrictions and enhance the efficacy and protection of image concealment. Hence, this research endeavors to confront the complications correlated with standard and neural image steganography by introducing a modern methodology that utilizes deep neural networks to accomplish efficient and secure image hiding. The recommended steganography procedure will be assessed regarding its effectiveness and safety using diverse datasets and performance criteria. The research objectives include:

1. Acquiring a complete comprehension of the art of steganography requires a thorough exploration of its basic principles and complexities, encompassing traditional techniques and the newest deep learning-based methodologies.
2. To venture into the depths of *CNNs*' potential within the realm of steganography and ascertain the optimal network structures and approaches to train for concealing images.
3. We aim to enhance the effectiveness of an already existing model of [15] for concealing and revealing images, making the process more efficient.
4. To offer illuminations and suggestions for future enhancements in the suggested steganography approach, which may involve possible advancements in network design, instructional techniques, or supplementary security measures.

## 1.3 Why Using *CNN*

*CNNs* with autoencoders for steganography, instead of depending only on LSB or other neural network architectures, we have the opportunity to get the best of the two worlds, security and efficiency. However, similar models like DenseNets, R-*CNNs* or FC-DenseNets can be used in their place although they might not be

that effective for use in steganography. For instance, DenseNets might be great for the purpose of image classification, but it will still need a lot of computational resources that can be hard to provide to real-time applications. R-*CNNs* and FC-DenseNets, although effective for object detection and segmentation, do not necessarily preserve the exact pixel data of input images which is very important for steganography where even a slight change can make the hidden data detectable or corrupt. *CNNs* are particularly skilled to capture and learn hierarchical image features, which are key to ensuring output images quality without disturbing the embedded secret message's perceptibility. Autoencoders, by learning optimal data representations and compressions, are a great tool for the encoder and decoder parts of steganography. This combination makes it possible to have a more advanced data embedding that is inherently more immune to common steganalysis techniques than LSB, which just changes the least significant bits and is prone to detection through simple statistical analysis. This technique not only strengthens the system against detection but also expands the embeddable data and answers some of the key limitations of the LSB method and others in the field of steganography. What is more, *CNN* with autoencoder is the best choice because they can carry and keep the encoded data safely and at the same time do not change the look of the initial image.

## 1.4 Thesis structure

Our thesis, titled "Enhanced CNN Approaches for Multi-Image Embedding in Image Steganography", is intelligently designed as follows: It starts with **Chapter 1: Introduction**, where it outlines the main idea of the research, clarifies the research challenge and the objective of the study and introduces the reasons for relying on Convolutional Neural Networks (*CNN*) approach, among others, to solve the problem. **Chapter 2:** In the section Literature Review, there is a complete overview of steganography with image and neural network steganography which are pointed out in academic works referred in various languages. **Chapter 3:** Dataset details Data sets used, such as ImageNet and Cifar10, may be applied. **Chapter 4:** Methodology follows, describing methods we use for our development of our architecture, including encoder, decoder, and full model. **Chapter 5:** Result Analysis states a summary of results for both of the two and three imagery applications. Finally, Chapter 6: Finally, the conclusions reveal what we found, the importance of our research, and what possible additional work is to be done.

# Chapter 2

## Literature Review

### 2.1 Background

#### 2.1.1 Image Steganography

Images are frequently employed in steganography as both cover and secret objects. A vast array of image file formats are present in the digital image domain, each having its own unique use case. Various steganographic algorithms are available for these diverse image file formats [2].

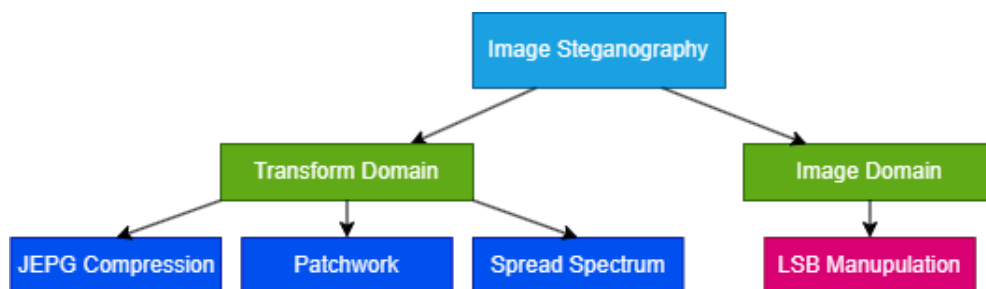


Figure 2.1: Simple Classification of Image Steganography Methods Proposed by [2]

Steganographic approaches for images may be separated into two primary groups: those that function in the Image Domain and those that function in the Transform Domain. In the Image Domain techniques, also referred to as spatial techniques, messages are hidden within the pixel intensity directly. On the other hand, Transform Domain techniques, also referred to as frequency techniques, first transform the images before embedding the message in them [2]. The messages can be any kind of data like text, image, audio, or video.

Although the work referenced in [2] did not explore neural image steganography, we can still make assumptions based on the definitions of image domain and transform domain. It is possible for neural image steganography to incorporate both techniques. One approach is to use neural networks to learn the mapping between cover images and stego images, embedding hidden information directly within pixel values. Another option is to utilize neural networks to manipulate transform domain coefficients, concealing information within the image.

According to [3], steganography methods that alter image files to conceal data comprise the subsequent techniques,

- Spatial domain steganography: Spatial domain steganographic technique involves a collection of uncomplicated strategies that generate a secret communication channel within the sections of the original image that are less susceptible to detection by the *HVS* [3].
- Transform Domain Techniques: While working in the time domain may be effective in some cases, utilizing the frequency domain to embed data in signals is typically more robust. It's worth mentioning that a lot of the most efficient steganographic systems currently being employed function within the transform domain.
- Spread spectrum: The technique of spread spectrum transmission is utilized in radio communications for transmitting messages at frequencies that are lower than the ambient noise level. Introducing pseudo-noise into cover images or treating them as noise can both be achieved by using spread spectrum in combination with steganography.
- Statistical methods: These techniques, also referred to as model-based techniques, have a tendency to alter the statistical characteristics of an image while still preserving them during the embedding process. This alteration is usually minor, allowing it to exploit the human tendency to overlook small changes in brightness.
- Distortion techniques: During the decoding process of distortion techniques, it is necessary to have knowledge of the original cover image. The decoder's role is to compare the distorted cover image with the original one and detect any differences to recover the secret message. Conversely, the encoder adds a series of modifications to the cover image. As a result, information is conveyed through signal distortion.

Neural network-based steganography utilizes a range of techniques including Transform Domain Techniques, Spread Spectrum, Statistical Methods, and Distortion Techniques, depending on the approach employed.

### 2.1.2 Neural Networks

A neural network is a computational model that tries to mimic the human brain's structure and functionality by using interconnected processing units called neurons to process and analyze complex data which are fed into the neurons as input. The firing pattern of neurons is learned through training, where the network adjusts the strengths of connections between neurons based on example data. A neural network is organized with layers, beginning with an information layer that gets crude information, trailed by at least one concealed layer for halfway handling, and a yield layer that conveys the last outcomes. The neuron connections, which are generally depicted as weights, are fundamental in determining the network's outcome. The network is trained so that it can regulate these weights and reduce the variance



between the projected and desired output. The backpropagation algorithm is implemented to perform this optimization process, which calculates the gradients of the network's error with respect to each weight. Neural networks possess the capability to grasp intricate patterns and deliver precise predictions or classifications from input data, which is the fundamental core of their strength. They display exceptional proficiency across several domains like image and speech recognition, natural language processing, and recommendation systems. With their revolutionary impact, they have transformed many sectors, including computer vision, machine translation, and medical diagnosis, among others [1] [19].

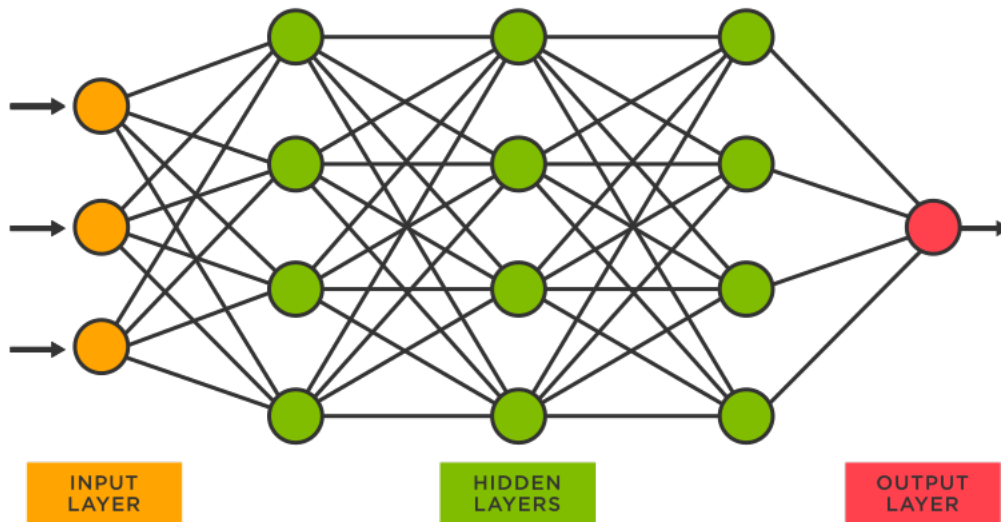


Figure 2.2: Simple Neural Network Structure [24]

## 2.2 Related Works

In this segment, we will be concentrating on performing a crucial evaluation of past significant research in the area of image steganography, with a particular emphasis on utilizing neural networks. We will scrutinize the different techniques applied, the architectural blueprints employed, and the outcomes achieved. Additionally, we will appraise the limitations of these investigations, accentuating the predicaments encountered in the image steganography field with neural networks, such as maintaining image quality, robustness against detection algorithms, image size, security, and computational intricacy.

According to [10], in our modern world, security of our data is probably one of the most important thing to us. The cover image is the one where the secret message is encoded. After encoding the final output is the stego-image. There are two types of steganalysis which are specific steganalysis and universal steganalysis. Specific steganalysis is only used with JPEG formats and the universal one is used for any formats. Steganography cover image formats are also discussed in this paper . JPEG format is the most commonly used image format in the world.Lossy compression is used in this format which doesn't allow anyone to edit the image repeatedly. BMP

format is found in compressed and uncompressed format. A 8 bit bitmap has upto 256 colors per pixel. RGB is also present here in other bit formats. PNG format is used when we need the image to stay at its original size and quality. This is normally used all over the internet and it retains its quality. TIFF format can hold a number of images in a single file. The image will not change its resolution when it's compressed in this format .It is perfect if you need to edit a digital image.Apart from this, Different types of color models for image formats. Some methods of image steganography are mentioned in this paper. Distortion method is mostly used in JPEG format.The secret data is encoded into the distortion of the image.LSB method is the most commonly used method. It changes the LSB of a pixel of an image to encode the secret data into the cover image. Other known methods are also included in this paper which can be helpful.

According to [14] development of CNNs boosted both image steganography security and multi-image embedding effectiveness. This kind of method is flawed besides other techniques such as the least significant bit (LSB) approach to steganography, but with the help of deep learning techniques, novel steganographic systems can be developed capable of withstanding steganalysis and the like. Their techniques allow the integration of a large number of clandestine images into an equally covert base picture while providing high visualization fidelity and accurate identification of the concealed pictures. The main problems stay in the area of balancing payload capability with concealment, but research here is continuing apace in general, and justifiably claiming better security in future comm technologies.

According to [23] a type of steganography is the data hiding in the image where it is very difficult for the hackers to go through the images and find out the hidden information. This method is very effective, especially in protecting sensitive data, something that forever addresses some of the holder's major issues on data protection. As the need for enhanced security of information under transmission increases, researchers have looked forward to finding a secure method of transmitting information without leakage and steganography is among the most important solutions. In this paper, they aim to present two methods for data hiding in the image domain. The first mechanism is the LSB technique which helps embed information bits within the said digital images. This is an insecure method because the attackers can easily guess the position and retrieve the hidden data which makes image steganography insecure even though it is well known. To enhance security, we propose a second technique: The R-Color Channel encoding that is enhanced with the RSA algorithm. This method utilizes the red color channel to conceal the information bits and other following bits are encoded for the RGB pixel of the original image to enhance the security. Based on the findings of this particular paper, it is evident that LSB and RSA qualified as two of the most popular algorithms used in image steganography have been shown to be effective in the process of protecting hidden data.

This study presents [5] a technique for safely concealing sensitive information in an image. The Arnold Transformation and the Mid Position Value Technique are the two main methods of operation. The first method, known as the Arnold Transformation, disrupts the way that pixels are typically arranged by jumbling up the image's data bits. Because of this, it is harder for someone to understand the secret data. The secret image is embedded within the main image using the second technique,

known as the Mid Position Value Technique. Additionally, it assists in creating the key needed to decode the hidden data. The inverse Arnold Transformation is used to undo the scrambling process. The pixels are put back in their original order by this method. The trial results are encouraging because the stego images closely resemble the original cover images and the image quality has been kept. With the use of this technique, more secret data can be concealed more effectively.

In the research work [9], the scholars put forth a novel approach to image steganography through the employment of a U-Net architecture that leverages deep learning. This method entails compressing and disseminating the information pertaining to the concealed image across all available bits in the cover image, thereby effectively addressing the issue of conspicuous visual cues. Furthermore, this technique also enhances the embedding capacity. With this method, they could achieve less error and higher accuracy, and higher payload capacity than *LSB* methods. However, the aforementioned model lacks the ability to exhibit any form of concealment that is competent in concealing multiple images within a singular cover image. Furthermore, the usage of u-Net is prone to various types of attacks, including adversarial and steganalysis attacks. The ramifications of such vulnerabilities could potentially jeopardize the security of the steganographic system and ultimately lead to a breach of confidentiality pertaining to the concealed data.

The intellectuals behind the publication [13] put forth a universal mechanism of image steganography that employs the framework of auto-encoding networks based on end-to-end trained deep Convolutional Neural Networks in order to guarantee the process of concealment and extraction. The educated network comprises two sub-networks, one for hiding deployed by the sender to encode a color image in another of the same size. The other network is used for extraction and is operated by the receiver to retrieve the secret image from the received stego image. The decoder network has been formulated as a complex network that takes the cover image and the secret image as input, both of which are separated into a 6-channel tensor. It is comprised of a duo of steps. The revealed network takes the stego image created by the encoding network as input. To create the confidential image result, a set of 3 x 3 convolutions were employed on the image, with each convolution accompanied by a BN operation for training acceleration and a ReLU activation function. In the final convolutional layer, a Sigmoid activation function was utilized to compress the convoluted feature channels into 3-channel features. The authors' inability to test their method on large images did not hinder satisfactory experimental results in terms of image imperceptibility and similarity to the original images.

The paper [7], introduces StegNet, a novel method that combines deep convolutional neural network techniques with image-into-image steganography. It establishes end-to-end mappings between the cover image and the embedded image, as well as between the hidden image and the decoded image. The proposed approach achieves a decoding rate of 98.2% or 23.57 *bpp* by modifying just 0.76% of the cover image on average, demonstrating a high payload capacity. The embedded image remains robust against statistical analysis. However, the paper lacks a comparison with other state-of-the-art steganography methods and fails to address the computational complexity and training time required for their approach.

In the proposed methodology by [18], they modified the *CNN* structure and a gain function of [4] based on multiple image similarity metrics was employed to maximize the undetectability between the cover and steganographic images. The proposed approach was assessed for its effectiveness by using commonly used image metrics such as *SSIM*, *MSE*, and *PSNR*, which makes it improbable for an AI detection tool to generate. The outcomes suggest that the steganographic images produced utilizing the suggested method are inconceivable to identify by the unaided eye, yet they still preserve a substantial amount of recoverability. The network they proposed has performed similarly to other steganography methods that exist and has shown a significant improvement in terms of *SSIM* as compared to the original [4]’s approach. The current research [18], as referenced by the scholarly [20] publication, did not delve into the aspect of steganalysis nor did it bring to light the crucial matter of the potentially larger size of secret images in comparison to regular images, posing a significant threat to the integrity of the secret image and the entire steganography procedure.

The paper [6] addresses the problem of image steganography, aiming to hide information within a cover image while maximizing payload capacity and maintaining robustness. It proposes a method that combines deep convolutional neural network techniques with image-into-image steganography. The encoding and decoding processes involve two nearly identical neural network structures, enabling effective modeling of high-level features in the latent space. The proposed method achieves a payload capacity of up to 23.57 bits per pixel by modifying only 0.76% of the cover image. Evaluation against traditional steganography analysis algorithms demonstrates its robustness. However, the paper has limitations, such as the evaluation being limited to a few datasets and lacking a comparison with other state-of-the-art methods in terms of both capacity and robustness.

The steganographic scheme suggested in [12] employs the Fully Convolutional Dense Connection Network (FC-DenseNet) to address the limited steganographic capacity of conventional methods that modify the carrier image. It hides a secret image within the carrier image, resulting in a stego-image that can be reconstructed to reveal the secret image. The scheme demonstrates a high *PSNR* and *SSIM*, enabling large-capacity image steganography with an average payload capacity of 23.96 bits per pixel. However, it’s worth noting that the scheme relies on FC-DenseNet, a deep learning model, which may necessitate significant computational resources and time for training and implementation. Moreover, due to the encoding and decoding time involved, the proposed scheme may not be suitable for real-time applications.

The work [20], delves into three distinct deep-learning network architectures (*CNN*, U-Net structure, and Swin Transformer structure) that address the image embedding and extraction issue in image steganography. By using the same dataset to validate the efficacy of the three networks, the authors illustrate the seamless integration of secret information with cover images but in the revealed images some noise points can be seen in the smooth areas. However, the Swin Transformer has exhibited inferior performance in reducing noise and preserving high and low-frequency information, indicating unpromising results for concealing secret information in images.

When implementing vividly colored images, the U-Net network proved to be the superior choice in terms of producing visually appealing stego images. Conversely, in the case of white cover images, the Swin network architecture outperformed the other two structures and achieved the most fulfilling outcomes. It is worth noting that this research work neglected to divulge any insights into the realm of steganalysis or the resulting dimensions of the image post-encoding with the secret image.

According to [8] The method of placing the data into one single picture has its own problems with steganography and its security. That is why to overcome with such problems the method of steganography that allows several images to hide information is improved. They suggested a new method of slicing secret data and spreading it among the nominees and cover pictures, thus increasing the level of file protection. What this means is that by being able to use many cover images for an image, we are essentially trying to safeguard data that is very crucial by making it almost impossible for intruders to decode the information without the keys to decryption. This work made the following contributions: A new payload distribution method in conjunction with image metadata as well as an authentication key technique; A method for image-based steganography for more practical purposes, namely towards secure data transmission. To this end, we use a batch steganography technique for sequential data hashing, making it more secure than if performed individually, as it minimises the chances of identifying hidden information. Moreover, in terms of usability and deployment, the HIs are expanded in order to achieve a more efficient user interface design. This involves encoding and decoding of the data to be transmitted for security, the retrieval of cover and payload image files payload files compression using the zip tool and division of bits to every image by the hashing algorithm. We use a value generator in storing the payload bits where each bit is stored in a specific serial number of an image, and image hashing with passwords as a way of making the distribution of the bits random thus making it difficult to decipher the images. This form of encryption and decryption ensures that the secret data being transferred from the source is safe and when decrypted at the destination the cover images are safely recovered.

This paper [17] discusses image steganography methods based on deep learning and highlights traditional, CNN-based, and GAN-based techniques. The review is conducted on methodologies, trends, and challenges for understanding researchers based on the current landscape; the methods in the first and second categories get their underpinnings from substitution techniques of LSB and deep convolutional networks, respectively. On the other hand, the third one, the GAN-based way, capitalizes on Generative Adversarial Networks for more security and capacity to hide information. The paper has highlighted the superior performance of GAN-based methods, especially CycleGAN. Some of the drawbacks include no benchmark datasets. The paper would be helpful in updating the knowledge base of and guiding scholars through these complexities of image steganography by compiling the present trends and future directions.

Another research work [16], which focuses on the difficulty of enhancing the ability to conceal information without raising suspicion is the main topic of the study. To do this, the authors suggest a cutting-edge strategy known as *ISN*. They want to hide additional information from observers in a way that doesn't stand out. The *ISN* framework views steganography and the recovery of hidden images as inverse prob-

lems in which the goal is to accurately retrieve information that has been accurately embedded into images. Within the same network, the authors present both forward and backward propagation processes. Backward propagation is used to adjust the network's weights depending on the prediction error, whereas forward propagation includes feeding input data through the network to make predictions. The authors effectively produce both the hidden images and the revealed photos with good quality by sharing all the parameters within the *ISN* network. By adding more channels to the hidden image branch, the *ISN* design increases steganography's capability. The findings show that using the *ISN* technique, 3-5 images may be successfully hidden and revealed while still having satisfactory container image quality. The concealed photos are successfully and faithfully restored. This demonstrates how well the suggested *ISN* approach works to increase the payload capacity of image steganography.

In [11] the authors proposed a new Steganography Convolution Neural Network model, which solves the problem of two images embedded in a carrier image and can effectively reconstruct two secret images. In this model, the preparation network is removed, and three different hidden networks ordinary *CNN*, Unet, and FCDenseNert are used to train the entire network. Before training the network, they uniformly adjusted the images input to the network to a size of  $256 \times 256$ . A 9-channel feature map is obtained after the carrier image and two secret images are concatenated. They output the stego image from this after a series of operations such as convolution, pooling, and concatenation through the hidden network. To reconstruct the two secret images in the extraction network they used convolution and batch normalization operations. They also compared the three hidden networks through different methods and found that FCDenseNert had the best outcome followed by Unet and then ordinary *CNN*. The experiments show that they were successful and also their model was suitable for almost all color image steganography, including remote-sensing images and aerial images.

The research work [15] describes a revolutionary method of "Multi-Image Steganography Using Deep Neural Networks" that makes use of neural networks to encrypt and decrypt a number of hidden images. This strategy uses CONV2D and ReLU layers to capture intricate visual patterns and characteristics as opposed to conventional methods, which rely on the *LSB* for encoding lower-resolution images. The prep network, hidden network, and reveal network are the three networks that make up the proposed architecture. Each network specializes in encoding or decoding a certain type of image. The Adam optimizer method is used by researchers to optimize the neural network. They use the TinyImageDataset, a collection of images with a 64x64 pixel size, for their studies. The Keras library was used to create the prep and hiding networks' shared architecture, which uses the same technique. The loss connected to the hidden image components is taken into account during training. The experimental findings show that three hidden photos were successfully concealed. However, a little less loss is seen when attempting to conceal two images. They discovered that the loss grows as more photos are obscured. The number of photos that must be concealed in cover images in order to get the desired results is not a criterion that the researchers set.

Based on the findings of [4] and [15], [25] has developed a superior model that outperforms in every metric, including *SSIM* and *PSNR*. This new methodology has the capability to hide four secret images within a single cover image. Although [15] used [4]’s methodology, they were only able to conceal three secret images. To attain such an outcome, a DeepCNN-based autoencoder was utilized to extract spatiotemporal characteristics from images and conceal four images within a solitary image, while ensuring size parity. The achievement of such a feat was made possible through this methodology. However, this research did not take into account steganalysis or the size of the hidden image, like many other previous works.

From the above discussion, it is observed that most of the image steganography techniques focused on error minimization or prevention from steganalysis or maximizing the number of secret images that can be embedded in the cover image. While doing it they only focused on one aspect. So in our work, we endeavour to tackle the distinct challenges that pertain to steganography, with the aim of augmenting its efficacy and security. We are aware of the importance of minimizing errors during image encoding and decoding, as well as the necessity of reducing the cover image size while concealing secret images. Our final goal is to develop a steganography technique that is secure and resistant to steganalysis methods. By concentrating on these aspects, we seek to establish an innovative and strong approach to steganography.

# Chapter 3

## Dataset Description

### 3.1 Description of the Data

#### 3.1.1 ImageNet

In our comprehensive exploration of the Tiny ImageNet dataset, our primary goal was to augment the versatility of our model by strategically selecting 10,000 samples across 200 distinct classes of data. This deliberate sampling aimed to ensure a diverse and representative dataset. The dataset was meticulously divided into training and testing sets, comprising 70% and 30%, respectively, of the overall data. Each image was initially standardized to a resolution of 64x64 pixels, providing a consistent input format for our model.

To facilitate effective training, we further organized the data into three distinct subsets: S1, S2, and C. Each subset contained 1,750 images and represented batches for secret image 1, secret image 2, and cover image, respectively. During training, this segmentation helped our model to identify as well as understand the characteristics of each group. We used the same methodology while testing as well. This actually generates three subsets which are Test S1, Test S2, and Test C alongside each with 750 photos. These subsets served as critical evaluation benchmarks, assessing the model's proficiency in accurately encoding the secret images into the cover image and decoding the secret images from the cover images.

Throughout the experimentation phase, we systematically varied the data counts where we range it from 1,000 to 5,000 samples. However, the observed results fell short of our expectations, prompting a meticulous reevaluation of our dataset strategy. Subsequently, we made the strategic decision to utilize 10,000 data points for both the training and testing phases. This revision was driven by the need to improve the whole model's performance and effectiveness in the complicated tasks of encoding and decoding hidden images within the cover images. The rational choice of this big dataset size was to get the right balance between the model complexity and the richness of the training data, which in the end led to the improved capabilities of our neural network in dealing with various and unknown image situations.



### 3.1.2 Cifar10

The CIFAR-10 is a widely used dataset in computer vision and machine learning research studies. It is a data set that comprises of 60,000 colour images of size 32x32 pixels which are divided into 10 classes with each class having 6000 images. The dataset is composed of 50,000 training images and 10,000 test images. The categories are aeroplanes, cars, birds, cats, deer, dogs, frogs, horses, ships and trucks, thus, they provide the required diversity of shapes and subjects for the identification purposes. Our model which was the first to be trained with the ImageNet dataset—a bigger collection of images that is used to improve deep learning methods, was tested with the CIFAR-10 dataset as test data. Specifically, we applied 10,000 images from the test part of CIFAR-10 to measure the model's generalization skills over different types of visual content, therefore, making it robust and precise in the real world. This way allows us not to get stuck in one experiment but to do a large number of experiments under different scenarios while at the same time considering CIFAR-10's compactness and difficulty to correctly validating the model.

# Chapter 4

## Methodology

### 4.1 Working Process

The inclusion of Convolutional Neural Networks (*CNN*) is a key part of our neural network structure in the proposed approach. *CNN* has shown great mastery in the field of image processing, which makes them the perfect fit for our upcoming project. These very specialized neural networks are very good at recognizing the complex structures and the spatial arrangements in images, thus they can easily identify the tiny details which are very important for steganography. Our model largely depends on Convolutional Neural Networks (*CNN*) to do both the encoding and decoding processes. *CNN*'s are utilised for their ability to effectively capture and represent complex visual information. The convolutional layers in the network aid in extracting and abstracting features, which helps in hiding information within the cover image and later reconstructing it during the decoding phase. The intentional incorporation of *CNN*'s highlights our commitment to utilizing advanced deep learning methods to enhance the robustness and effectiveness of our proposed steganographic approach.

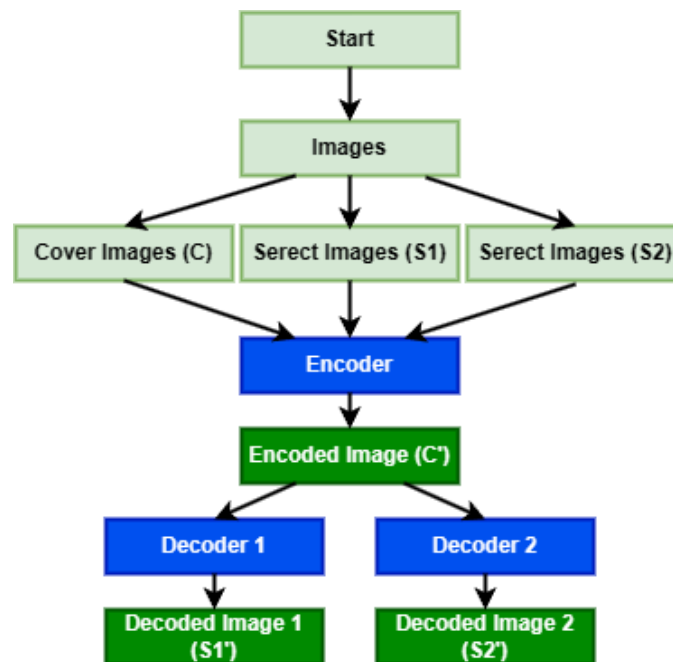


Figure 4.1: Working Process of Our Full Model for 2 Image Steganography

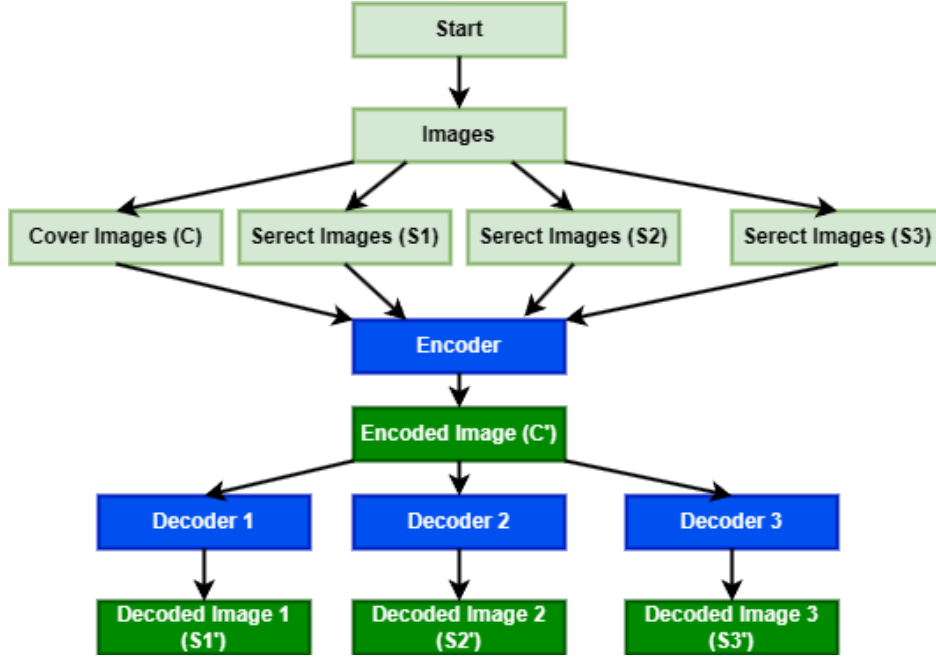


Figure 4.2: Working Process of Our Full Model for 3 Image Steganography

## 4.2 Description of the Model

Our model architecture consists of two essential components: the Hiding Network (Encoder Model) and the Reveal Network (Decoder 1 Model, Decoder 2 Model for 2 Image Steganography, Decoder 1 Model, Decoder 2 Model, and Decoder 3 Model for 3 Image Steganography) as seen in figure 4.1 and 4.2. The primary goal of this model is to encode the information pertaining to the secret images, which are S1, S2 and S3 (only for 3 Image Steganography), into the cover image C, which results in a modified cover image called C'. The encoded image C' is precisely constructed to resemble the original cover image C, as it provides the undetectability of the embedded information. The Reveal Network utilises the data. It creates the decoded secret images S1', S2', and S3' (only for 3 Image Steganography) from C'. The goal is to decode the process and then produce S1', S2' and S3' (only for 3 Image Steganography) which accurately represent the original secret images S1, S2 and S3 (only for 3 Image Steganography). The model's approach includes both encoding and decoding, which forms the foundation of our steganographic technique. This shows the seamless integration of hiding and retrieving data while maintaining the visual integrity of the source photos.

### 4.2.1 Encoder Model

#### 4.2.1.1 For Two Image Steganography

The encoder model utilizes a structure of a convolutional network at multiple scales. It initiates with three parallel input layers, each undergoing separate initial convolutions with varying kernel sizes of 3x3, 4x4, and 5x5. These kernel sizes facilitate the extraction of features at different scales, with smaller kernels capturing finer details and larger ones encapsulating broader features. The outcomes of these convolutions are then combined to merge the diverse feature maps into a cohesive representation.

This process is repeated in a second preparatory phase, augmenting the complexity of the feature extraction. Consequently, the outcomes derived from these two stages, in conjunction with an additional input which is the real cover image  $C$ , are subsequently merged and propagated through a sequence of supplementary convolutional layers. The network augments its filter count to 128, indicating a deepening to capture more intricate and abstract representations of the input data. The final combined output is then fed into a final convolutional layer to generate the encoded output, which is a condensed representation of the original input, ready for further processing in subsequent components of the model.

Our model demonstrates improvements when compared to the approach presented by [15], particularly in the encoder, where the architecture can be seen in 4.3. These enhancements can be attributed to the augmented architecture, which integrates additional convolutional layers and increased complexity. There are multiple convolutional layers in this preparation network in case of the both hidden photos. In addition, the dimensions of the filters are  $3 \times 3$ ,  $4 \times 4$ , and  $5 \times 5$  which enables the acquisition of a wide spectrum of hierarchical data. These highly improvements allow the model to extract more complex spatial information and patterns from the input photos. Also, the hiding system shows improved complexity and depth by using many convolutional layers with various filter sizes such as  $3 \times 3$ ,  $4 \times 4$ , and  $5 \times 5$ . The augmented depth, coupled with higher kernel sizes, enables the retrieval of more complex and comprehensive contextual characteristics, so increasing the model's efficacy in multi-image steganography assignments.

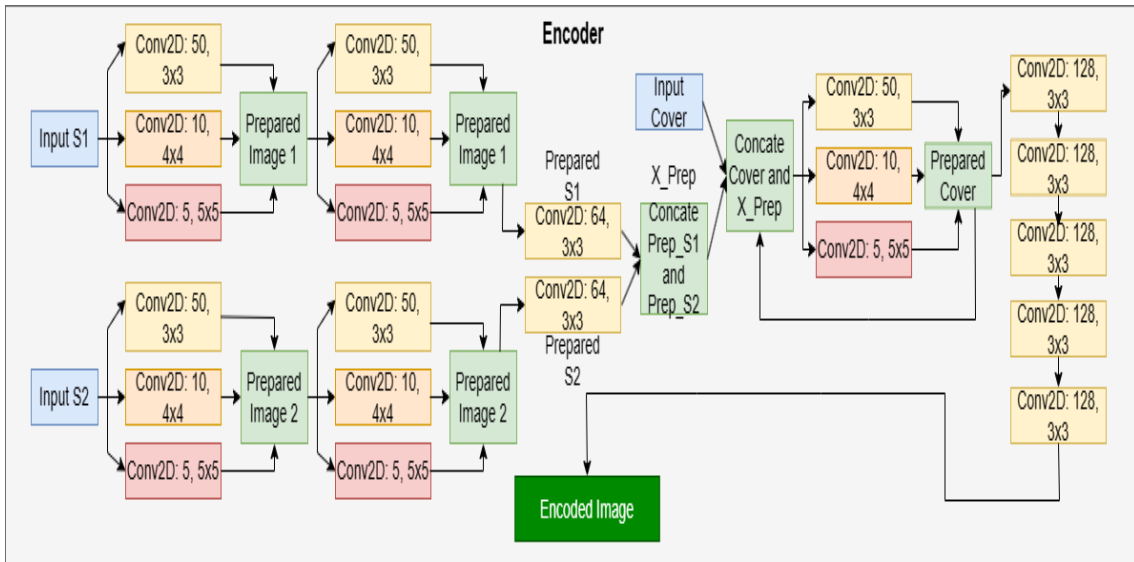


Figure 4.3: Encoder Model for 2 Image Steganography

#### 4.2.1.2 For Three Image Steganography

In our exploration of 3 Image Steganography, we initially extended the 2-image encoder model by integrating an additional input and preparatory layer for a third secret image, employing the same convolutional approach with varying kernel sizes ( $3 \times 3$ ,  $4 \times 4$ , and  $5 \times 5$ ) for nuanced feature extraction as we can see in figure 4.4. This setup was modified from our original architecture, which combined these features

with a real cover image and processed them through an augmented series of convolutional layers with increased depth and filter counts up to 128, aimed at capturing more intricate representations.

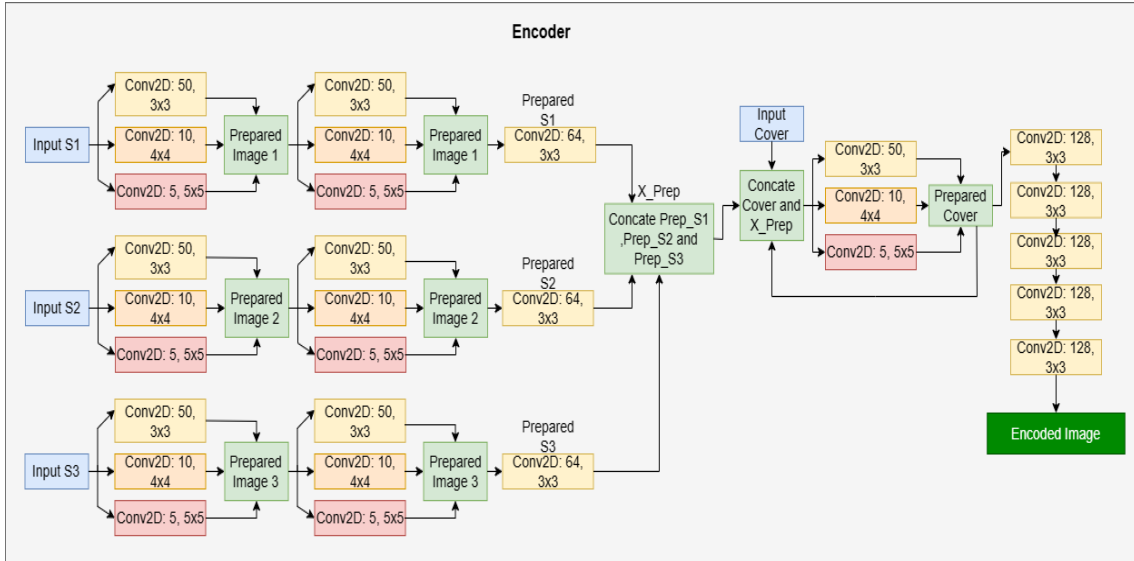


Figure 4.4: Encoder Model for Three Image Steganography, Similar to Two Image Steganography

The new encoder, as seen in figure 4.5, for steganography with 3-image, is designed modularly with the `conv_block` function serving as its core that facilitates the building of convolutional layers using a standard format. Thus, the approach leads to a more mobile and scalable architecture, hence ignorance of the specificity of the old encoder architecture's individual layers where all construction needs to be done manually. In particular, the new encoder consists of 9 preparation layers (three layers per secret image) and 12 hiding layers, settled in the framework of loops which provide the model with flexibility and scalability. Every `conv_block` will appear twice, in both the preparation and hiding layers, and is composed of 3x3, 4x4, and 5x5 kernels to ensure feature extraction is done. The lean code approach, which incorporates data on-demand smart interfaces, reduces maintenance time, speeds up change, and makes `DataType` agnostic. On the other hand, the old encoder has 33 layers (18 in preparation and 15 in hiding) which makes it less efficient. The multiplicity of layer count has been significantly reduced, and the specific architectural structure of Encoder 2 leads to less computation enabling efficient learning, which is especially important for the variety of datasets. This layout does, however, result in the improvement of model performance by the fact of performance being faster and the errors which are more likely to be the source of complex configurations being reduced.

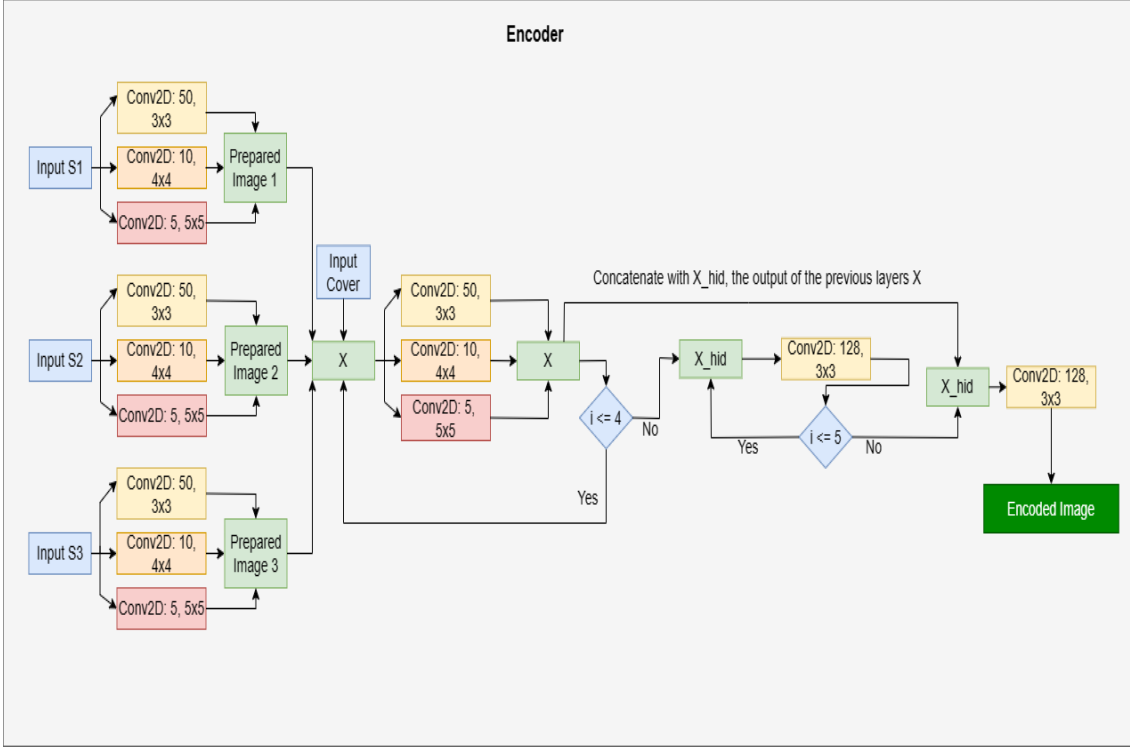


Figure 4.5: Final Encoder Model for Three Image Steganography

## Encoder Algorithm Overview

### 1. Parallel Input Processing:

- The encoder starts with three input layers that are running in parallel, each of them getting one of the secret images ( $S_1$ ,  $S_2$ ,  $S_3$ ). Thus, the processing of each image is carried out at the same time and independently, thereby, the features from one image will not contaminate the initial stages of processing for the other images.

### 2. Multi-Scale Convolutional Layers:

- For each secret image, a series of convolutional layers with different kernel sizes ( $3 \times 3$ ,  $4 \times 4$ , and  $5 \times 5$ ) is used. This strategy is designed to capture a diverse range of features. This strategy is designed to capture a diverse range of features:
  - **3x3 Convolutional Layers:** These layers try to gather the smallest and the most intricate details and the most exact features of the images.
  - **4x4 Convolutional Layers:** These thus represent the equilibrium between the extreme resolutions and the high-level structures since the former preserves the medium-scale features which are of utmost importance for keeping the image data as a whole.
  - **5x5 Convolutional Layers:** The spotlight should be on the general characteristics of an image, thus the model will be able to understand the overall context and the bigger patterns in each picture.

### 3. Feature Integration and Augmentation:

- Through iterative attention of various convolutional layers for each image, the outputs are combined into one, complete feature map. This integration allows the mixing of fine and contextual information, thus, making the data depth more advanced and useful for embedding.
- The integrated feature maps for each image go through a second iteration of similar multi-scale convolutions, thus, the complexity of the feature maps increases and the robustness of the extracted features takes its form.

#### 4. Concatenation with Cover Image:

- Moreover, the cover image is examined through its own set of convolutional layers for the purpose of joining it with the secret images. This procedure guarantees that the cover image is able to effectively include and hide the feature maps of the secret images.
- The feature maps that are the result of the processing of the three secret images and the cover image are then merged, thus forming a complex and multifaceted feature representation.

#### 5. Deep Convolutional Network:

- The concatenated feature maps are finally sent to a more advanced system of convolutional layers with an increasing number of filters (from 50 to 128). These layers are intended to filter the combined data, to abstract the higher-level features and to even more hide the information that is embedded in the intricate patterns of the cover image.

#### 6. Encoded Output Generation:

- At the end of the encoding process, there is a set of convolutional layers that gradually squeeze the data into a single encoded image. This hidden image with the encoded information is the cover picture of the innocuous image, and this makes the hidden data invisible and will not be noticed easily if there is not a lot of computation to decode it.

### 4.2.2 Decoder Model

The architecture model of Decoder 1, Decoder 2 and Decoder 3 is the same as seen in the figure 4.6, that is, they have the same architecture version. The purpose of Decoder 1 is to show  $S1'$ , which code is image 1 ( $S1'$ ). However, Decoder 2 is assigned to making  $S2$  the image that is not apparent while also translating  $S2$  as the hidden image. Equally, Decoder 3 has derived the same architectural blueprint as the other decoders, namely, Decoder 1 and Decoder 2, with the sole function of extracting the hidden image 3 for the 3-image steganography ( $S3'$ ), which is based on the same input structure as the other two decoders. The design of constructive methods assures simultaneous stylistic working in both frameworks. The start of the decoder model's architecture is an input layer that carries Gaussian noise along with it, and it is believed that the noise will enhance the capability of the network to manage small variations in the input data or noise in the input data. The architecture is based on multi-scale convolution with filters of  $3 \times 3$ ,  $4 \times 4$ , and  $5 \times 5$ ,

each of which is used with the input data, including noise. The main purpose of these convolutions is to highlight a wide range of features that are already present in the image. Instead of depending on filters with one specific dimension to detect or get information about the patterns and complexities, the network becomes able of capturing them at various scales. Past the first confusion layer and concatenation, the network further includes an additional 64 filters to reach the final output layer. This is a more compact version (as compared to the work of [15]) that has fewer layers and a pathway for input to directly reach output. In [15] they suggest a more intricate design of the decoder which includes an interweaving of different filter sizes in a convolution and concatenation process. Complicatedness cannot raise performance, only heightens the level and number of difficulties. Some happen when a model achieves a good architecture and is simple enough to stand a chance of running as effectively as other models that do not explore new data. Our decoder model's architecture limits the amount of features that may be extracted at each layer, which reduces the possibility of overfitting. This could potentially result in better results in terms of kernel and filter sizes. Overfitting is a common problem in complex models where the network becomes excessively customized to the training data, thus hindering its capacity to generalize to unfamiliar data. All three models utilize the multi-scale convolution technique to capture features at different levels of abstraction. However, our model achieves this more effectively by processing the input through a shorter sequence of transformations. This results in a more resilient feature extraction procedure, as each layer just enhances the features without introducing unnecessary intricacy that might obscure the fundamental patterns in the data.

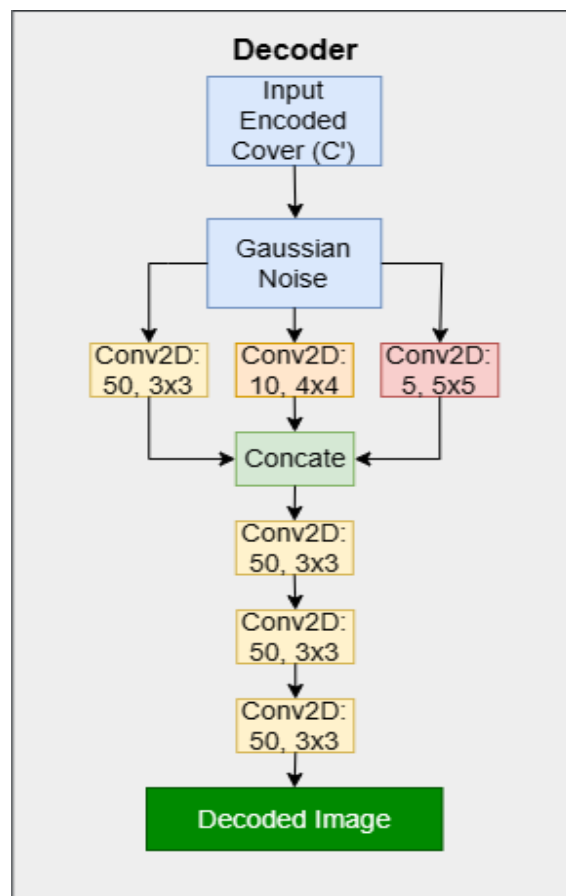




Figure 4.6: Decoder Model

## Decoder Algorithm Overview

- **Input Reception:**

- The decoder gets the cover image  $C'$  which is redesigned to hide the information within its pixel structure during the encoding process.

- **Gaussian Noise Application:**

- The Gaussian noise is introduced into the encoded image at the beginning of the decoding process. This stage is designed to mimic the environmental changes and the subsequent perturbations as well as to check the robustness of the decoder, thus, confirming the stability and reliability of the decoder in different situations.

- **Multi-Scale Convolutional Processing:**

- The noise image is at the same time processed through three different convolutional layers, each using a different kernel size, which are  $3 \times 3$ ,  $4 \times 4$ , and  $5 \times 5$ .
- $3 \times 3$  Convolutional Layer: The process concentrates on the outing of the very small details and the high-frequency components which are the basis of the precise image features.
- $4 \times 4$  Convolutional Layer: The medium-scale features are captured and the integration of these features are made easier so that more features can be integrated.
- $5 \times 5$  Convolutional Layer: The layer is designed to analyze the larger image patterns and the broader features which are necessary to understand the general image context and the structure.

- **Feature Map Concatenation:**

- The three convolutional layers produce outputs which are then merged to create one feature map. This aggregation of varied feature scales thus combines different feature scales and thus adds to the features that are used for the decoding stages.

- **Deep Convolutional Refinement:**

- The feature map formed by the consecutive (concatenated)  $3 \times 3$  convolutional layers goes through another round of processing. This sequential deep convolutional method's filters and details the features in a way that produces more accuracy than the previous method of spying on people, on the other hand, what the hidden content is by a feature.
- These layers successively modify the features, thereby, the model is able to restore the original images with the accuracy of the encoded data from the iterations.

- **Final Image Reconstruction:**

- The processed feature map is finally passed to the last convolutional layer, which reforms the decoded image. This picture aims to be a replica of the original images before encoding, thus, it succeeds in showing the hidden data with high quality.

### 4.2.3 Full Model

Our full model is a complex neural network architecture comprising an encoder and two decoders. This structure suggests a multi-task learning framework where the encoder learns to encode input data into a condensed representation, and each decoder is responsible for reconstructing the secret images  $S_1'$ ,  $S_2'$  and  $S_3'$  respectively.

The data flow can be described as follows:

1. **Input Preparation:** The model takes three separate inputs, `input_S1`, `input_S2`, `input_S3`, and `input_C`. These represent different images for secret images 1, 2, 3 and cover.
2. **Encoding:** These inputs are fed into the `make_encoder` function which constructs the encoder part of the model. The encoder processes the inputs and generates a new representation of the cover image, denoted as `output_Cprime`. This encoded output is expected to contain the essential information from all three inputs, compressed into a more efficient form.
3. **Decoding:** The encoded representation `output_Cprime` is then passed to two separate decoders constructed by `make_decoder1`, `make_decoder2` and `make_decoder3`. These decoders are designed to reconstruct the original inputs ( $S_1$ ,  $S_2$  and  $S_3$ ) from the encoded representation. `decoder1`, `decoder2` and `decoder3` are set to non-trainable, suggesting that they have been pre-trained and are now being used in a fixed manner, to guide the encoder's training by providing a stable target.
4. **Loss Functions:** The loss functions used in the model are crucial for guiding the training process. Two main components are considered:

(a) **Reveal Network Loss (`rev_loss`):**

The reveal network loss measures the dissimilarity between the true secret image ( $S$ ) and the predicted secret image ( $S'$ ) generated by the reveal network. It is calculated using the mean squared error term multiplied by a hyperparameter  $\beta$ :

$$\text{rev\_loss}(s_{\text{true}}, s_{\text{pred}}) = \beta \cdot \|S - S'\|^2$$

Here,  $\beta$  controls the importance of the reconstruction error in the overall loss.

(b) **Full Model Loss (`full_loss`):**

The full model loss comprises three components:

i. **Cover Image Reconstruction Loss:**

This term measures the difference between the true cover image ( $C$ ) and the predicted cover image ( $C'$ ). It aims to ensure accurate reconstruction of the cover image.

ii. **Secret 1 Reconstruction Loss:**

This term assesses the dissimilarity between the true secret 1 image ( $S_1$ ) and its predicted counterpart ( $S'_1$ ). Similar to the reveal network loss, it uses the mean squared error term multiplied by  $\beta$ .

iii. **Secret 2 Reconstruction Loss:**

This term quantifies the difference between the true secret 2 image ( $S_2$ ) and its predicted counterpart ( $S'_2$ ), employing the mean squared error multiplied by  $\beta$ .

iv. **Secret 3 Reconstruction Loss:**

This term quantifies the difference between the true secret 3 image ( $S_3$ ) and its predicted counterpart ( $S'_3$ ), employing the mean squared error multiplied by  $\beta$ .

The overall full model loss is a combination of these three components:

$$\text{full\_loss}(y_{\text{true}}, y_{\text{pred}}) = \|C - C'\|^2 + \beta \cdot \|S_1 - S'_1\|^2 + \beta \cdot \|S_2 - S'_2\|^2 + \beta \cdot \|S_3 - S'_3\|^2$$

The loss guides the training process, driving the model to minimize discrepancies between the true and predicted values for cover and secret images while considering the importance of secrets through the  $\beta$  parameter.

5. **Model Compilation:** The decoders are compiled with adam optimizer and a loss function named `rev_loss`.
6. **Autoencoder Construction:** The outputs from all decoders, along with the encoded representation, are concatenated into a single tensor and used as the output of a new model, `autoencoder`. This model is then compiled with an optimizer and a loss function, `full_loss`, which is designed to measure the performance of the full architecture, likely considering both the quality of reconstruction and the efficiency of encoding.

The training process for such a model typically proceeds in batches as follows:

1. **Batch Processing:** A batch of data is prepared for each of the three inputs. The size of these batches must be consistent across all inputs since they are processed in parallel through the encoder.
2. **Forward Pass:** During each training iteration, a forward pass is performed where the batched data is fed through the encoder and then through each decoder. This generates a set of outputs that can be compared against targets to compute loss.
3. **Backpropagation:** The loss computed from the output of the autoencoder is backpropagated through the network to update the weights of the encoder. Since the decoders are frozen (non-trainable), their weights remain unchanged, and only the encoder is updated. This means the encoder learns to produce representations that allow the fixed decoders to reconstruct the data accurately.

The autoencoder model is a type of neural network that aims to learn a compressed representation of the data, which can then be used to reconstruct the original data. It has two main parts:

- **Encoder:** Maps the input data to a compressed representation.
- **Decoder:** Attempts to reconstruct the input data from the compressed representation.

In training an autoencoder, the goal is to minimize the difference between the original data and the reconstructed data, thereby forcing the encoder to learn a useful representation. The combination of encoder and decoder parts allows the model to be trained end-to-end. In this specific setup, like figure 4.7 and 4.8, the encoders and decoders may be trained separately first, with the encoders fine-tuned later in the context of the full autoencoder model, using the outputs of the fixed decoders as a guide for reconstruction accuracy.

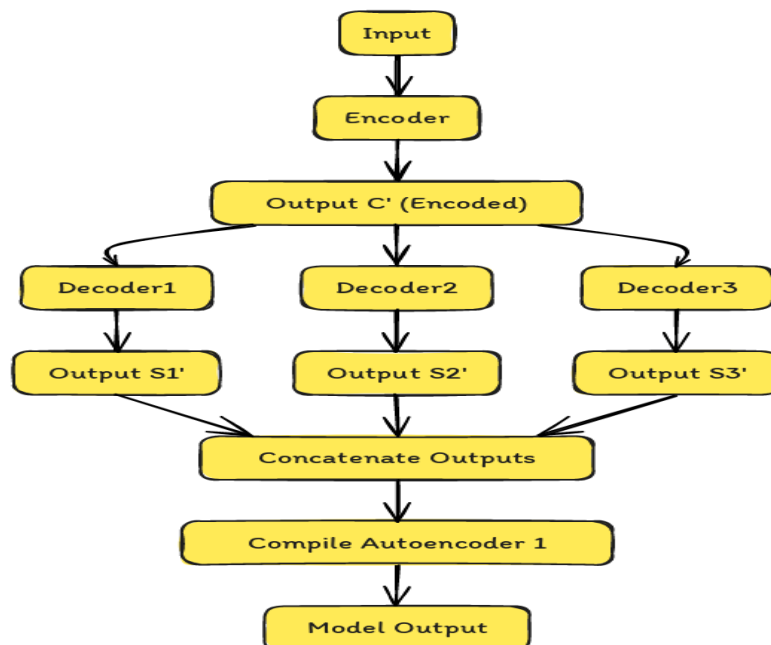


Figure 4.7: Full Model

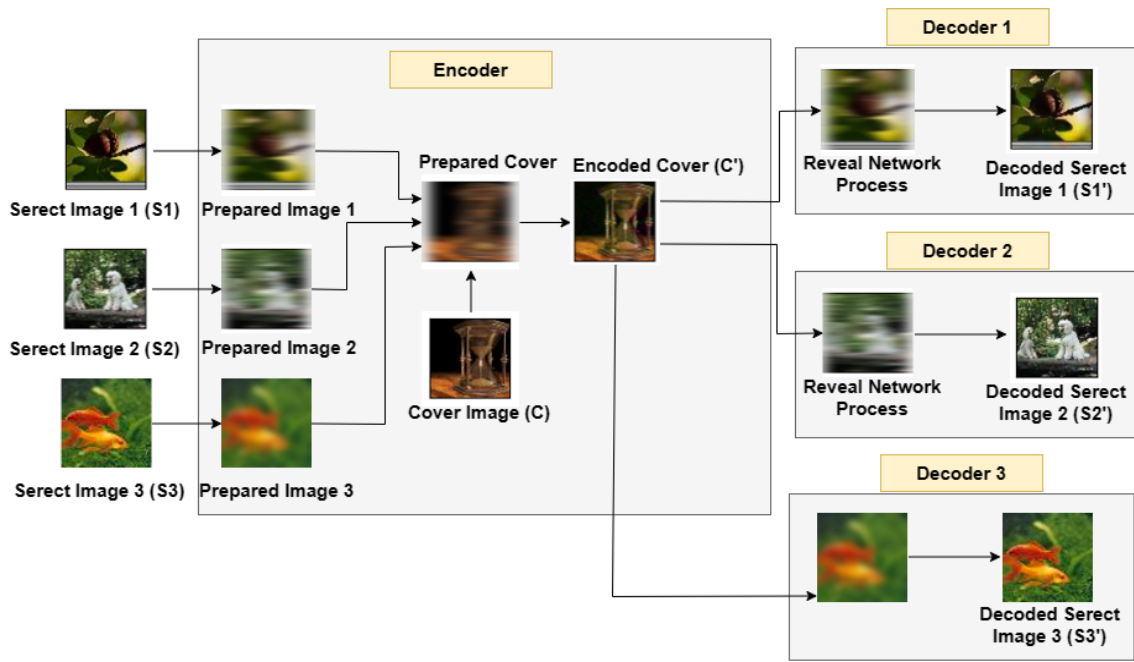


Figure 4.8: Full Model Workflow

# Chapter 5

## Result Analysis

In our preliminary analysis, we extensively explored various architectural configurations and layer setups for our encoder-decoder model. Through systematic experimentation, we identified the current architecture as the most effective, achieving superior performance compared to numerous alternatives. Notably, our approach involved training the model with different activation functions, such as ReLU, Leaky ReLU, and Tanh, revealing that a diverse set of activation functions contributes to the model's flexibility and overall effectiveness. This stands in contrast to previous work, which predominantly utilized ReLU in all their models. The mathematical equations of ReLU, Leaky ReLU, and Tanh are,

- **ReLU:**

$$f(x) = \max(0, x) \quad (5.1)$$

- **Leaky ReLU:**

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases} \quad (5.2)$$

- **Tanh:**

$$f(x) = \tanh(x) \quad (5.3)$$

### 5.1 Two Image Steganography

Table 5.1: Error Per Pixel

Activation	Model	S1'	S2'	C'
Relu	Training [15]'s Model	12.1705885	10.796396	17.404426
	Testing [15]'s Model	11.944164	10.54237	17.170265
	Training (Proposed Model)	11.1674385	12.168935	18.223452
	Testing (Proposed Model)	11.0192995	12.344757	18.069624
TanH	Training (Proposed Model)	69.883194	72.48468	160.75792
	Testing (Proposed Model)	67.62504	73.122055	163.91539
Lrelu	Training (Proposed Model)	10.181644	9.602057	16.44105
	Testing (Proposed Model)	10.266071	9.496834	16.29391

Table 5.2: Accuracy Per Pixel

Activation	Model	S1'	S2'	C'
Relu	Training [15]'s Model	94.42	94.79	94.42
	Testing [15]'s Model	94.48	94.86	94.47
	Training (Proposed Model)	96.84	96.52	94.59
	Testing (Proposed Model)	96.88	96.54	94.63
TanH	Training (Proposed Model)	76.98	75.80	43.36
	Testing (Proposed Model)	77.77	75.43	42.22
Lrelu	Training (Proposed Model)	97.18	97.26	95.12
	Testing (Proposed Model)	97.16	97.28	95.16

The performance metrics of the *CNN* model trained with different activation functions reveal interesting insights. When comparing the results of the Relu, TanH, and Leaky Relu activation functions, it becomes evident that the Leaky Relu (Lrelu) consistently outperforms the others across various aspects. In both training and testing phases, Lrelu exhibits lower errors per pixel for all three components (S1', S2', and C') compared to Relu and TanH and it also outperforms the previous model of [15] that solely relied on the Relu activation function. Moreover, Lrelu achieves higher accuracy per pixel, indicating better pixel-wise predictions. The robust performance of Lrelu can be attributed to its ability to handle vanishing gradient problems during training, which can be crucial in the convergence and learning processes. The Leaky Relu's slight slope for negative inputs allows for the flow of a small gradient during backpropagation, preventing dead neurons and promoting better learning. This flexibility seems to contribute to Lrelu's superior performance, making it a favourable choice for the given neural network architecture and dataset. It is worth noting that the Leaky Relu was implemented with an alpha value of 0.01 for both the encoder and decoder models. In the midst of our experimentation, we implemented L1 and L2 regularization techniques to mitigate overfitting in our model. Despite applying a regularization value of 0.01 for both L1 and L2 in the Leaky ReLU (Lrelu) model, the results were suboptimal. Consequently, we made an informed decision to retain the Lrelu model, as it consistently outperformed other alternatives even when subjected to regularization measures. This highlights the nuanced nature of model development, where a combination of architecture, activation functions, and regularization techniques must be carefully considered to achieve optimal performance.

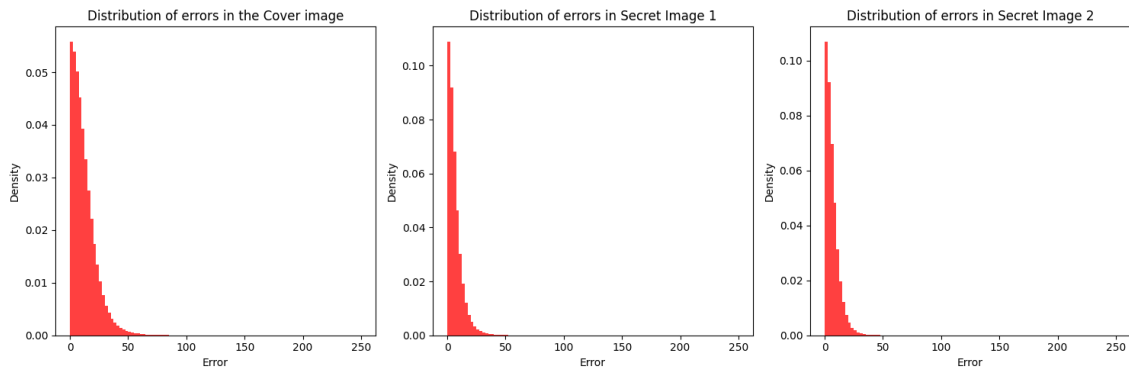


Figure 5.1: Distribution of Errors in Cover and Secret Images using Lrelu Activation

The histograms in 5.1 display the error distributions for a Cover image and two

Secret images. All distributions are left-skewed, indicating that most errors are small, with the Cover image showing a higher peak, suggesting fewer errors. Secret Images have broader error distributions, implying a slightly higher average error, but still, the majority of errors remain near zero, indicating an overall low error rate and effective encoding. Now, transitioning to the visual representation of our results, we will showcase images that further illustrate the performance and quality of our model’s reconstructions.

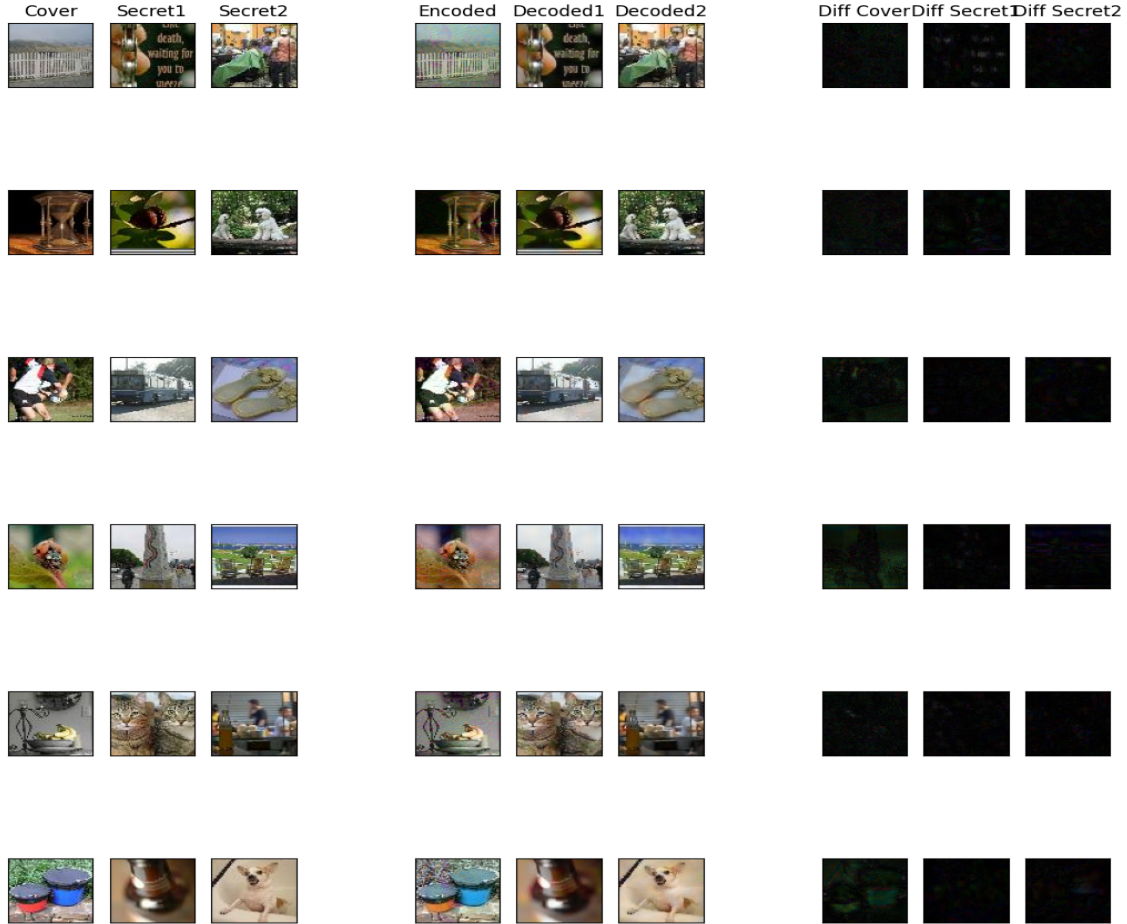


Figure 5.2: Result of Concealing Two Secret Images.

Figure 5.2 shows the results of our steganography model and the result image is arranged in a series of columns, each serving a specific purpose in the context of encoding and decoding images, presumably for a steganographic application. The difference images (Diff Cover, Diff Secret1, and Diff Secret2) are generated using the equation:

$$\begin{aligned}
 \text{diff\_S1} &= |\text{decoded\_S1} - \text{input\_S1}|, \\
 \text{diff\_S2} &= |\text{decoded\_S2} - \text{input\_S2}|, \\
 \text{diff\_C} &= |\text{decoded\_C} - \text{input\_C}|.
 \end{aligned}
 \tag{5.4}$$



Table 5.3: Performance Metrics of figure 5.2, Result of Concealing Two Secret Images and Comparison with [15]

Model	Image	Accuracy	EPP	SSIM	PSNR	MSE	RMSE
[15]'s Model	Cover	85.99	20.95	0.87	19.63	0.1079	0.3285
	S1	86.83	25.58	0.84	22.36	0.1780	0.4219
	S2	87.59	28.85	0.79	38.20	0.1257	0.3545
Proposed Model	Cover	95.93	12.90	0.94	29.69	0.00107	0.033
	S1	96.83	11.97	0.96	27.49	0.00178	0.042
	S2	97.51	8.27	0.99	32.46	0.00057	0.024
[15]'s Model	Cover	86.03	22.45	0.85	19.58	0.4107	0.6409
	S1	91.06	21.83	0.85	25.65	0.3108	0.5574
	S2	85.05	18.52	0.89	20.25	0.8956	0.9462
Proposed Model	Cover	95.48	14.28	0.84	28.28	0.00149	0.038
	S1	97.02	10.04	0.97	30.44	0.00090	0.030
	S2	97.74	7.33	0.99	32.14	0.00061	0.024
[15]'s Model	Cover	91.05	29.02	0.85	18.58	0.4586	0.6772
	S1	89.25	16.52	0.91	30.85	0.5823	0.7631
	S2	87.79	18.24	0.88	29.80	0.1285	0.3854
Proposed Model	Cover	95.28	15.23	0.97	28.87	0.00130	0.036
	S1	97.99	6.53	0.98	33.53	0.00044	0.0209
	S2	97.53	8.05	0.96	32.62	0.00055	0.023
[15]'s Model	Cover	89.48	25.80	0.87	20.85	0.2048	0.4525
	S1	88.40	27.77	0.82	29.87	0.1789	0.4229
	S2	87.99	16.85	0.88	31.02	0.5849	0.7649
Proposed Model	Cover	93.49	20.52	0.91	26.91	0.00204	0.045
	S1	97.96	7.08	0.98	32.97	0.00051	0.022
	S2	96.95	10.27	0.98	31.87	0.00065	0.025
[15]'s Model	Cover	84.80	25.58	0.79	17.52	0.8851	0.9408
	S1	89.66	20.85	0.77	28.58	0.8548	0.9245
	S2	91.58	10.58	0.95	30.58	0.5784	0.4605
Proposed Model	Cover	96.48	11.42	0.95	30.17	0.00096	0.031
	S1	97.60	7.99	0.98	31.90	0.00065	0.025
	S2	98.03	6.58	0.98	34.21	0.00038	0.019
[15]'s Model	Cover	91.52	11.52	0.93	21.25	0.9985	0.9992
	S1	89.98	9.25	0.89	30.78	0.0254	0.1593
	S2	93.89	11.95	0.89	29.25	0.0025	0.05
Proposed Model	Cover	94.39	18.75	0.96	26.24	0.00238	0.049
	S1	98.04	6.34	0.98	35.20	0.00030	0.017
	S2	97.52	8.17	0.96	31.19	0.00076	0.027

Table 5.3 represents performance metrics for images encoded and decoded with our steganographic technique figure 5.2 and compare the results with [15]. Each row corresponds to a specific image category (Cover, S1, S2), and the columns present various evaluation metrics. In the following paragraphs, we will provide a detailed explanation of the metrics for the first three rows of the table, which corresponds to a batch of steganography processes. Our proposed model consistently outperforms the referenced model in terms of accuracy, SSIM, PSNR, MSE, and RMSE across

all tested images (Cover, S1, and S2). Notably, the proposed model achieves higher accuracy and SSIM, indicating better image quality and similarity, while maintaining lower MSE and RMSE values, demonstrating improved error performance.

Table 5.4: Overall Performance Comparison of Proposed Model With and Without Gaussian Noise

Model	Dataset	S1 Accuracy (%)	S1 EPP	S2 Accuracy (%)	S2 EPP	Cover Accuracy (%)	Cover EPP
Proposed Model with Noise	ImageNet	97.16	10.26	97.28	9.49	95.16	16.29
Proposed Model without Noise	ImageNet	97.28	10.06	98.09	6.72	94.30	21.04

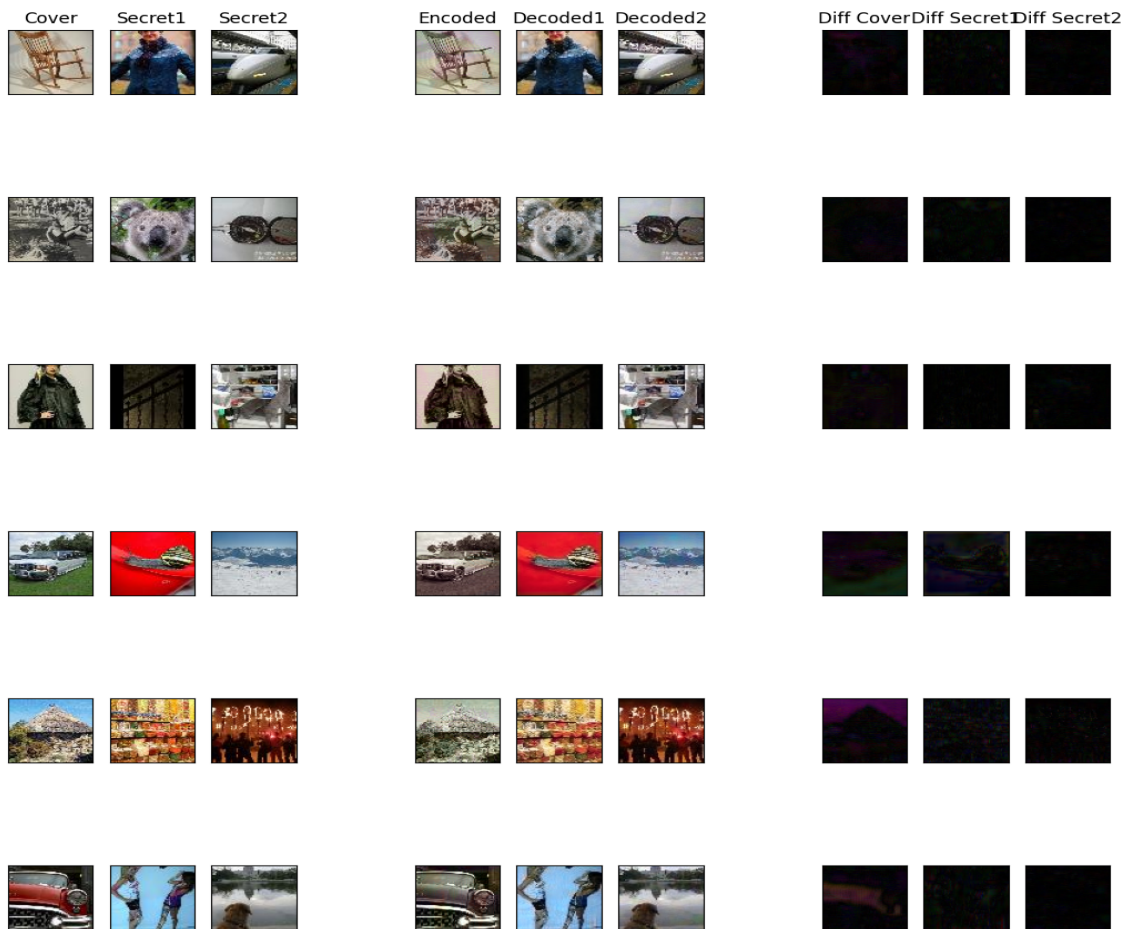


Figure 5.3: Result of Concealing Two Secret Images (Without Noise)

Table 5.5: Performance Metrics of figure 5.3, Result of Concealing Two Secret Images (Without Noise)

Image	Accuracy	EPP	SSIM	PSNR	MSE	RMSE
Cover	96.90	10.50	0.99	34.85	0.0003	0.0181
S1	97.70	7.58	0.98	33.24	0.0004	0.0218
S2	98.49	4.93	0.99	37.25	0.00057	0.0137
Cover	97.17	8.71	0.99	35.80	0.0002	0.0162
S1	97.56	8.12	0.98	32.60	0.0005	0.0234
S2	98.65	4.39	0.98	38.35	0.0001	0.0121
Cover	96.56	10.34	0.98	35.78	0.0002	0.0162
S1	98.73	4.21	0.95	36.27	0.0002	0.0154
S2	98.42	5.37	0.99	37.14	0.0001	0.0139
Cover	93.91	18.51	0.98	29.04	0.0012	0.0353
S1	92.99	21.88	0.95	28.47	0.0014	0.0377
S2	98.32	5.72	0.98	35.53	0.0002	0.0167
Cover	92.44	29.77	0.98	28.83	0.0013	0.0362
S1	95.41	14.98	0.97	28.26	0.0019	0.0386
S2	97.58	8.26	0.98	33.75	0.0004	0.0205
Cover	94.89	19.01	0.99	35.55	0.0002	0.0167
S1	96.91	10.19	0.96	31.48	0.00071	0.0266
S2	98.50	4.97	0.98	37.33	0.0001	0.0136

The observed discrepancies in the model’s performance with and without Gaussian noise can be elucidated by examining the respective impact on decoding and encoding phases, as highlighted by the results in the table 5.4. When Gaussian noise is included during decoding, we see slightly higher error rates (EPP) for secret images (S1 and S2) and the cover image in the model with noise compared to the one without. Specifically, for the model with noise, the S1 accuracy is 97.16% with an EPP of 10.26, S2 accuracy is 97.28% with an EPP of 9.49, and cover accuracy is 95.16% with an EPP of 16.29. Conversely, the model without noise shows improvements in these metrics with S1 accuracy at 97.28% and EPP at 10.06, S2 accuracy at 98.09% and EPP at 6.72, and cover accuracy at 94.30% but with a higher EPP of 21.04. From these findings, it can be concluded that when the Gaussian noise is filtered out, the decoded images in the decoding process are more precise and clear, supported by EPP which represents the error per pixel needed for proper decoding. It is because induced noise disturbances are not present during the decoding process which helps in getting the secret information with more accuracy and without any change. At the same time, this remains detrimental to redundancy and the model’s generalization ability in the encoding phase: When there is no noise, the EPP for the cover image is greatly increased. The reasons could be attributed to the fact that the addition of Gaussian noise during training causes the encoder network to adapt to variations of the data similar to the real world. Thus, while the raw decoding rate might be tweaked to appear superior when noise is not included, the overall steganographic capability of the embedding algorithm, especially in terms of cover’s EPP and concealment, suffers badly. This highlights the fact that steganographic systems always have a trade off between the decoded content and the ability of the

organization or the individual encoding the information to go unnoticed while at the same time ensuring that the contents of the ‘hidden’ information are not lost.

## 5.2 Three Image Steganography

Table 5.6: Performance Metrics for Concealing Multiple Secret Images

Model	Dataset	S1 Accuracy (%)	S1 EPP	S2 Accuracy (%)	S2 EPP	S3 Accuracy (%)	S3 EPP	Cover Accuracy (%)	Cover EPP
[15]’s Model	ImageNet	90.01	34.40	89.02	36.92	91.35	30.39	89.39	35.64
Initial Model	ImageNet	93.53	20.98	95.67	14.99	94.81	17.63	92.85	24.18
Proposed Model	ImageNet	96.21	12.96	96.16	13.11	95.64	15.04	93.53	21.73
Proposed Model	Cifer10	96.45	12.11	96.66	11.31	96.29	12.55	93.84	20.82
Proposed Model (Without Gaussian noise in Decoder)	ImageNet	96.39	12.33	94.82	17.16	94.76	17.76	92.84	25.65

Table 5.6 shows performance metrics for three models with steganography tasks which are measured in accuracy and error per pixel (EPP), on the three different secret images(S1,S2, S3) in relation to the cover image. Our First Model is a clear illustration of the fact that the secret images are processed with higher accuracy and much lower EPP than the model of [15]. Regardless, Our Final Model for the image net dataset continue this process and improves the accuracy and increase it from all metrics. For example, the accuracy for S1 is 96.45%, initially. 90.01% in [15]. Moreover, the upcoming sequential applications S2 through S3 indicate that our modified technique indeed enhances the image quality by minimizing the pixel level misplacement while still safeguarding the high-level concealment accuracy. This is a very significant achievement as it is very useful in applications that require robust steganography with minimal distortion and the images that are concealed remain both secure and intact after the encoding process. The sum of squared errors per pixel (SSE) or we are calling error per pixel (EPP) for an image  $I$  and its corresponding decoded image  $D$  can be calculated as:

$$\text{EPP} = \sqrt{\frac{1}{N} \sum_{i=1}^N (255 \cdot (I[i] - D[i]))^2}$$

Where:

- $N$  is the total number of pixels in the image.
- $I[i]$  represents the pixel value at position  $i$  in the original image.
- $D[i]$  represents the pixel value at position  $i$  in the decoded image.

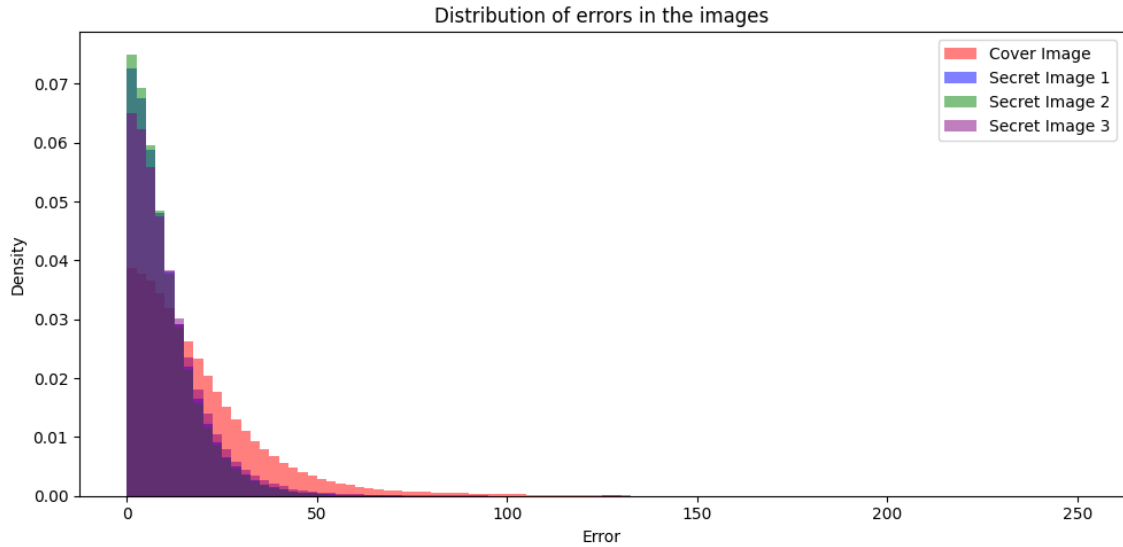


Figure 5.4: Distribution of Errors in Cover and Secret Images

We can see from the histogram 5.4 that the cover image (red) has a much wider dispersion of errors, but the density peaks sharply at the lower error values and quickly decreases, which means most of the pixels have minor errors. Secret Image 1 displayed a pattern with the nearest probability of a nature in the error application but with a few broader areas that create steeper peaks. Secret Image 3 showed the same behaviour, but its spread was narrow and steeper than the first one. The gravitational curve (magenta) provides a much more normal distribution, however, the distribution has the steepest slope close to zero error, thus, Secret Image 2 has the least error dispersion and the smallest error scale among all the secret images. These distributions demonstrate that the steganographic embedding and extraction processes are very effective for Secret Image 2 comparatively, which means that the encoding technique might be more efficient or the image content is less complex and so it is easier to retain the quality during the process. On the other hand, it is worth noting that encrypted images and respective secret images have higher and more stretched-out error distributions, which may be the reason that they have significant transformations from the original material. The effect could be that visual or data integrity is impacted.

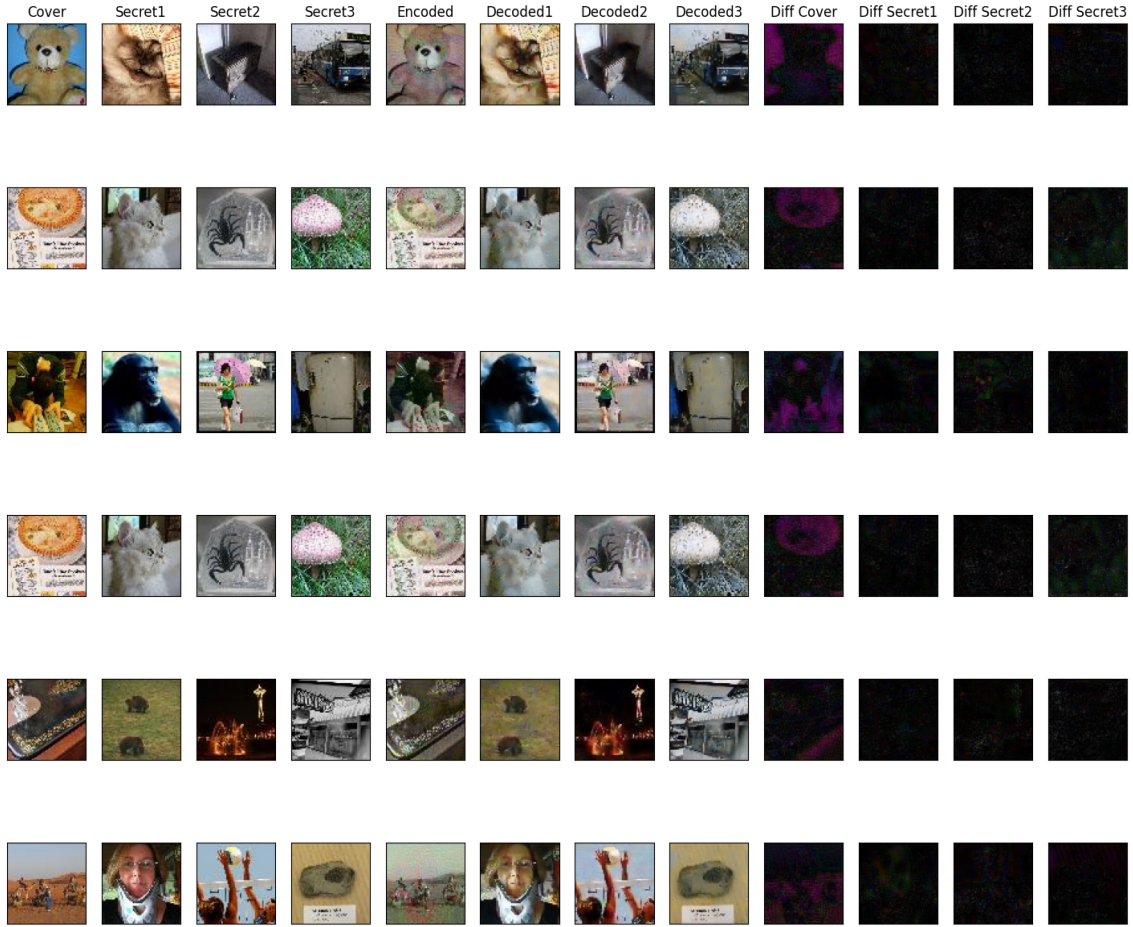


Figure 5.5: Result of Concealing Three Secret Images.

Figure 5.5 shows the results of our new steganography model with an enhanced encoder and the resulting image is arranged in a series of columns, each serving a specific purpose in the context of encoding and decoding images, presumably for a steganographic application.

Table 5.7 meanwhile, the metrics of the covering performance for hiding three secret images on top of a cover image, which are illustrated in figure 5.3, are listed in the table as follows. The metrics assessed include Accuracy(%), Error per pixel (EPP), Structural similarity index (SSIM), Peak signal-to-noise ratio (PSNR) and Mean squared error (MSE). Accuracy is a measure of the accuracy of prediction of total correctly predicted data points compared to the total. EPP quantifies the average error in pixels that help in finding the magnitude of the simulated concealment error. SSIM is a metric of visual impact that measures the extent to which the errors are structural information changes, with the higher values being the fewer errors and more similarity to the original. PSNR requires the next quality of images obtained after reconstruction relative to the initial one, which is measured by the ratio of maximum possible signal power to the power of corruption noise. The higher the value is the better the quality. Following this, MSE calculates the mean squared difference between original and encoded pictures serving as an accurate reconstruction error indicator. Taking an example from the table, for the 'Cover' image in the first row, the metrics are as follows: It is 94.11% accuracy, EPP is 19.60, SSIM is 0.93, PSNR is 27.79 and 0.00166 MSE. This demonstrates a decent

quality of concealment and quite a low level of palpability of the error. The balance in the image quality between the concealed image and the distortion and the noise of the image after the hidden stage is what characterizes the result as being good. The fact that the steganographic method can maintain image integrity while hiding the secret data shows its effectiveness. The table 5.7 also compares the performance of the images with our proposed model and with the work of [15] where our model is performing better in most of the metrics significantly specially SSIM.

Table 5.7: Performance Metrics for Concealing Three Secret Images

Model	Image	Accuracy (%)	EPP	SSIM	PSNR	MSE	RMSE
[15]'s Model	Cover	93.73	20.64	0.82	24.12	0.0038	0.0616
	S1	94.14	9.98	0.86	30.61	0.0008	0.0283
	S2	93.11	16.23	0.86	24.37	0.0036	0.06
	S3	95.42	12.12	0.84	27.50	0.0017	0.0412
Proposed Model	Cover	94.11	19.60	0.93	27.79	0.00166	0.0407
	S1	97.02	9.73	0.89	28.04	0.00148	0.0384
	S2	96.48	11.78	0.96	27.12	0.00194	0.0440
	S3	96.42	11.78	0.96	27.12	0.00194	0.0440
[15]'s Model	Cover	92.31	15.23	0.91	27.62	0.0017	0.0412
	S1	94.01	9.94	0.85	30.61	0.0008	0.0283
	S2	96.81	11.62	0.91	27.64	0.0017	0.0412
	S3	96.16	12.54	0.82	26.90	0.0020	0.0447
Proposed Model	Cover	94.09	12.76	0.96	27.26	0.001662	0.0407
	S1	94.02	12.74	0.93	26.95	0.00202	0.0449
	S2	96.46	12.30	0.86	26.40	0.00281	0.0530
	S3	96.24	12.30	0.86	26.40	0.00281	0.0530
[15]'s Model	Cover	94.26	18.78	0.88	24.91	0.0032	0.0566
	S1	93.47	14.83	0.85	28.77	0.0013	0.0361
	S2	94.10	19.45	0.92	23.17	0.0048	0.0693
	S3	93.55	15.49	0.83	26.16	0.0024	0.0490
Proposed Model	Cover	97.03	21.50	0.96	26.79	0.00210	0.0458
	S1	96.15	19.01	0.94	27.46	0.00182	0.0426
	S2	96.93	11.17	0.92	27.66	0.00172	0.0414
	S3	96.94	11.17	0.92	27.66	0.00172	0.0414
[15]'s Model	Cover	90.73	20.64	0.82	24.12	0.0038	0.0616
	S1	91.14	9.98	0.82	30.60	0.0008	0.0283
	S2	95.11	16.23	0.86	24.37	0.0036	0.06
	S3	96.42	12.12	0.84	27.50	0.0017	0.0413
Proposed Model	Cover	93.69	23.53	0.94	26.65	0.00229	0.0478
	S1	96.24	11.87	0.93	27.42	0.00183	0.0427
	S2	96.54	11.87	0.93	27.42	0.00183	0.0427
	S3	97.24	9.28	0.95	29.69	0.00107	0.0327
[15]'s Model	Cover	94.26	18.78	0.88	24.91	0.0032	0.0567
	S1	95.47	14.83	0.84	28.77	0.0013	0.0361
	S2	94.10	19.45	0.92	23.17	0.0048	0.0693
	S3	95.55	15.49	0.83	26.16	0.0024	0.0490
Proposed Model	Cover	94.11	19.60	0.93	27.79	0.01662	0.1289
	S1	97.00	9.73	0.89	30.09	0.00980	0.0989
	S2	97.13	10.30	0.83	28.48	0.01418	0.1190
	S3	96.49	11.78	0.96	27.12	0.01941	0.1393
[15]'s Model	Cover	93.01	24.00	0.80	24.70	0.0033	0.2089
	S1	95.70	14.45	0.84	26.6	0.0021	0.0458
	S2	94.93	17.33	0.84	23.94	0.0040	0.0632
	S3	95.14	16.65	0.83	24.99	0.0031	0.0556
Proposed Model	Cover	93.81	20.17	0.88	27.00	0.0019	0.0436
	S1	95.70 38	10.09	0.95	31.94	0.0006	0.0244
	S2	94.07	19.53	0.93	24.49	0.0035	0.0592
	S3	95.38	15.90	0.91	24.50	0.0035	0.0592



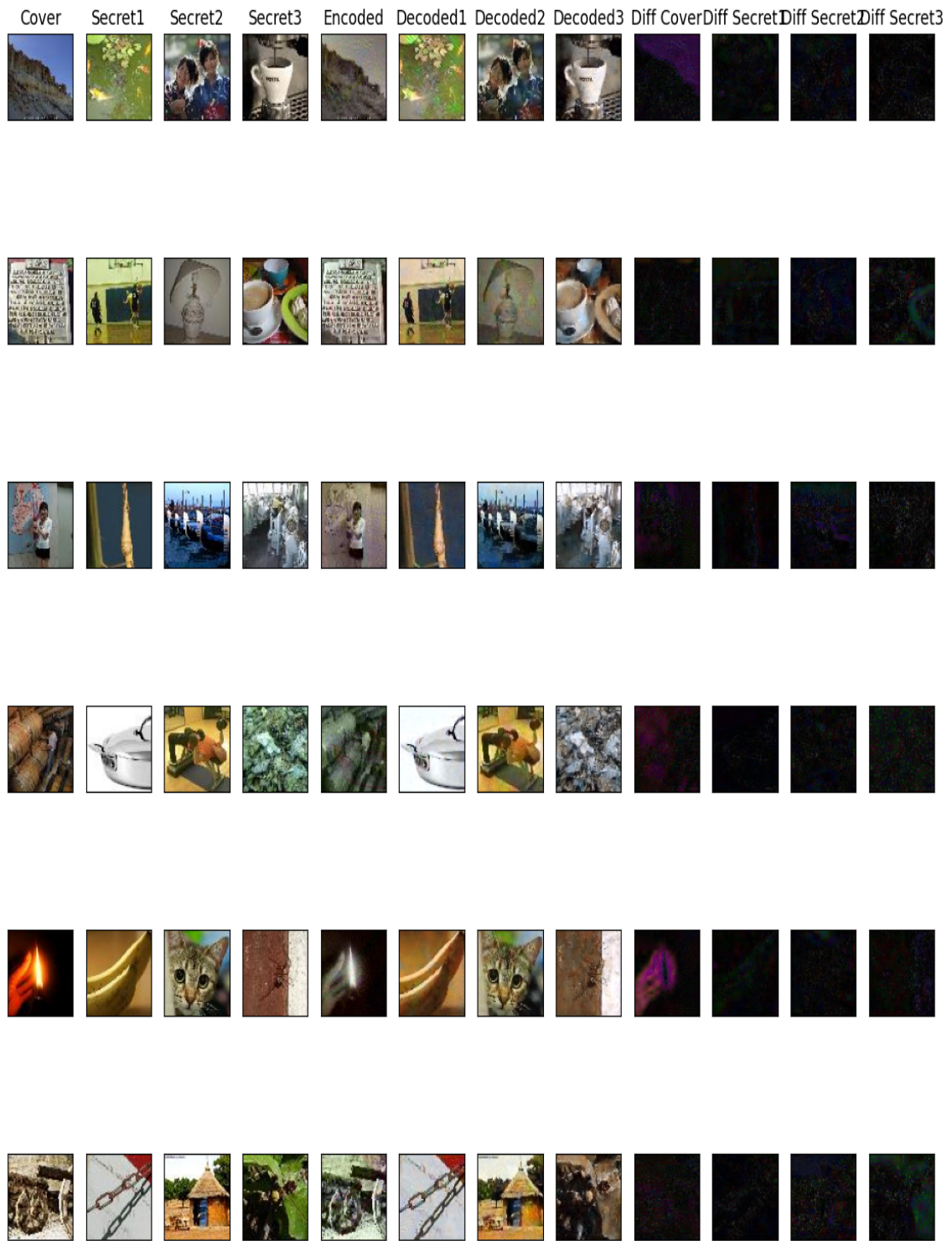


Figure 5.6: Result of Concealing Three Secret Images (Without Gaussian noise in the Decoder).

Table 5.8: Performance Metrics for Concealing Three Secret Images (Without Gaussian noise in the Decoder)

Image	Accuracy (%)	EPP	SSIM	PSNR	MSE	RMSE
Cover	91.17	30.29	0.85	25.45	0.002	0.044
S1	95.67	14.15	0.92	28.33	0.001	0.0316
S2	95.22	15.41	0.94	26.0	0.002	0.0447
S3	96.14	13.19	0.92	26.24	0.002	0.044
Cover	95.69	14.20	0.98	28.05	0.0015	0.0387
S1	95.20	15.18	0.94	28.23	0.0015	0.0387
S2	95.15	15.73	0.88	26.58	0.0021	0.0458
S3	94.10	19.91	0.88	24.2	0.0037	0.0608
Cover	93.81	20.17	0.88	27.00	0.001	0.0321
S1	97.07	10.09	0.95	31.94	0.0006	0.0245
S2	94.07	19.53	0.93	24.49	0.0035	0.1934
S3	95.38	15.90	0.91	24.51	0.0035	0.1934
Cover	92.50	24.33	0.89	26.51	0.0022	0.0469
S1	97.53	8.12	0.97	32.12	0.0006	0.0245
S2	95.62	14.13	0.93	26.91	0.0020	0.0447
S3	93.38	20.98	0.92	23.46	0.004	0.0632
Cover	90.41	38.89	0.89	29.06	0.0013	0.0361
S1	96.74	10.77	0.96	30.23	0.0009	0.03
S2	96.2	12.48	0.95	28.33	0.0014	0.0374
S3	95.62	14.23	0.86	27.01	0.0019	0.0436
Cover	94.08	18.91	0.94	25.11	0.0038	0.0617
S1	96.66	11.31	0.94	28.74	0.0013	0.0361
S2	93.43	20.76	0.90	23.07	0.0049	0.07
S3	92.84	23.43	0.85	23.13	0.0048	0.06928

By removing the Gaussian noise in the decoder of a steganography system, one observes that steganography especially when decoding is slightly decreased, as illustrated by the experiment that was done with the ImageNet dataset. For the model with Gaussian noise, we observe an accuracy of 96.21%, SSIM of 96.16%, PSNR of 15.04 dB, and an MSE of 93.53. In contrast, the model without Gaussian noise shows decreased performance with an accuracy of 96.39%, SSIM of 94.82%, PSNR of 17.76 dB, and an MSE of 92.84, along with a deterioration in a miscellaneous metric from 21.73 to 25.65. It is due to the various reasons indicated below that worsen and lead to this degradation. Gaussian noise acts as a type of regularization to limit the model’s amplification of noise-free conditions that were observed during training, which should improve the model’s performance in the new unseen data that may contain distortions. Furthermore, it also mimics some realistic circumstance where data can be slightly off which prepares the model on how it is going to handle and rectify it in the most efficient manner possible. Without this noise, the decoder may fail to learn robust features which are needed in reconstructing the secret image, thereby reducing its usability in real life situations where variations and degradations of the image are possible. This conclusion is supported visually and numerically in the table 5.8 and figure 5.6, showing a deteriorating of the model’s

stability and the ability to reconstruct the images when Gaussian noise is removed. The addition of Gaussian noise to the training process is an essential step because it helps the model not to be too rigid and unfavourable to a range of minor changes in data inputs during regular operation.

It should be noted that the research included situations like corner cases. In the example shown in figure 5.7 the cover frame is white and S1 is all-black, while S2 is a randomly generated image. Another edge case as displayed in figure 5.8, has a totally white cover page and S1 and S2 are all black. Even though the human eye will never be able to discern changes between the two covered layers, the model portrays outstanding accuracy of reading and interpreting such complex tasks.

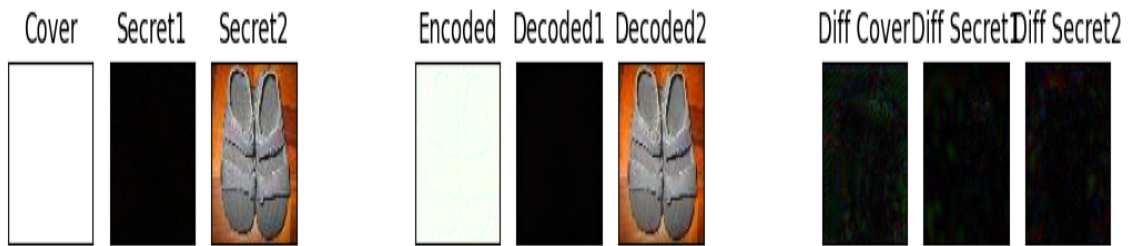


Figure 5.7: Result of Concealing Two Secret Images in Plain White Cover Image.

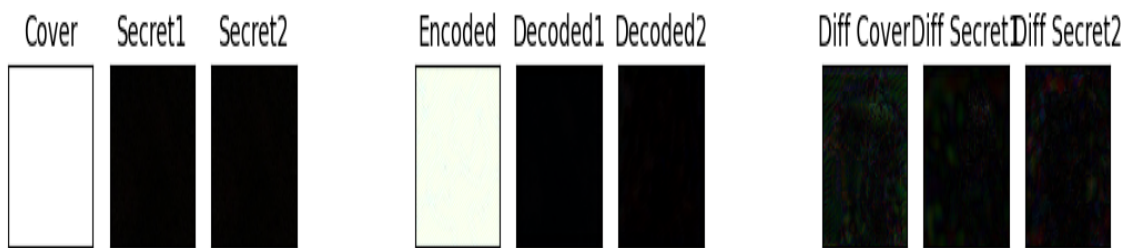


Figure 5.8: Result of Concealing Two Full Black Secret Image in Plain White Cover Image.

## Chapter 6

# Implementation and Analysis of Security Measures

Steganography has a functional difference from cryptography. In cryptography, there is always something hidden, which is the secret, and what follows is to ensure that no one gains access to the content of the secret. While steganography conceals the very presence of the secret by placing it within the context of ‘gathering of harmless or visible data’. Therefore even if an adversary gets a hold of the decoder architecture, it is still almost impossible for him to extract the hidden message. The image with the secret is transmitted over the internet, and it only takes a split second to post millions of images over the internet. This huge number means that anybody else could not single out and decipher the concealed message, even though full or partial decoder information is available. Let’s assume the adversary knows which image has some secret on it. Then, the generation of the encoder and decoder networks with different sizes of the convolution layer using Leaky ReLU activation makes it hard to establish patterns by observing the images through other network architectures. Furthermore, the encoder has to encode and integrate several secret images all at once, each of which passes through convolutions and concatenations in a sequence. If an adversary gets a partial part of the decoder architecture, they will work in partial knowledge of layers and parameters without trained weights and biases, missing customised loss functions, and training methodologies; nothing would be said about the context that the image underwent and the preprocessing steps it was subjected to, how it integrates multiple secret images, and the presence of Gaussian noise and robustness schemes. These factors, in a cumulative way, create significant difficulties in the attempts to reverse the steganographic process and decode the hidden images. As pointed out by [22], inherent security in steganographic methods mainly stems from the least obvious and most distinguished aspects of specialized implementations; thus, it is unlikely that an opponent will be able to penetrate without inside information. Moreover, the direct reliance on the encoder’s outputs as well as the importance of adaptive learning rates and training techniques in establishing network performance complicate the adversary’s job. It is therefore through these complex mechanisms and the impossibility of reproducing the original network, that the multi-image steganography system’s solid and secure basis against such attacks is provided. Based on our experiments, even if the adversary shares up to 75% similarities with the decoder network, he cannot access the secret images. Figure 6.1 shows the output of the up to 75% similarity with the decoder network after

removing the last two convolutional layers.

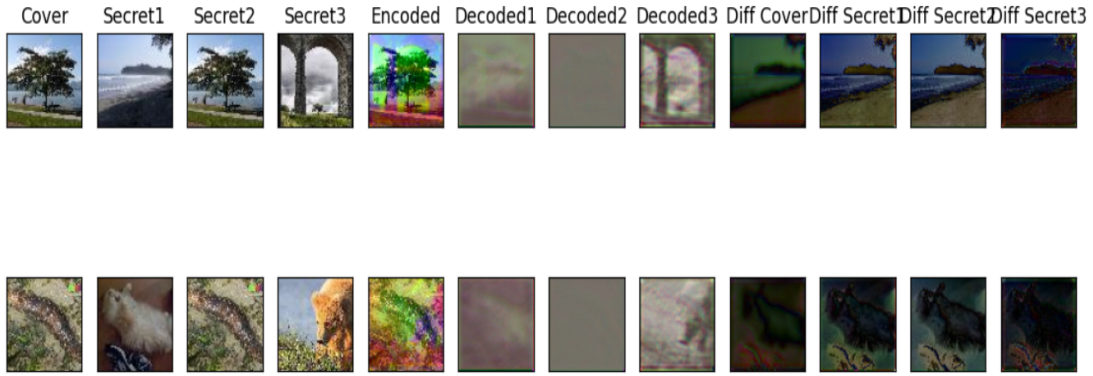


Figure 6.1: Result of Removing the Last Two Convolutional Layers from Decoder

## 6.1 Implementation of Cryptographic Algorithms in Our Model

We further implemented AES (Advanced Encryption Standard) and ChaCha20, in the sense that the images used for training the encoder and the decoder population are encrypted before the training exercise is initiated. The AES is one of the most famous symmetric cryptography algorithms because of its high operating speed and security levels, which requires and processes the keys with a length of 128 bits and works only with the fixed amount of data blocks. ChaCha20 – Another fast-stream cypher designed by Daniel J. Bernstein to provide high security when encrypting data using a pseudorandom keystream. The workflow, as illustrated in the figure ?? encompassing cover images C and secret images S1, S2, and S3, encrypts these images through AES or ChaCha20 encryption before passing to the encoder. The encoder combines the secret images with the cover image to produce the encoded image (C'), is then required to pass through three decoders. Every decoder obtains one or more of the secret images (S1', S2', S3') and each of the images is decrypted to reconstruct the original S1, S2, and S3. The additional layer mentioned above greatly increases the overall resistance against adversarial attacks as it incorporates the advantages of both cryptography and steganography methods to secure and encrypt the data successfully.

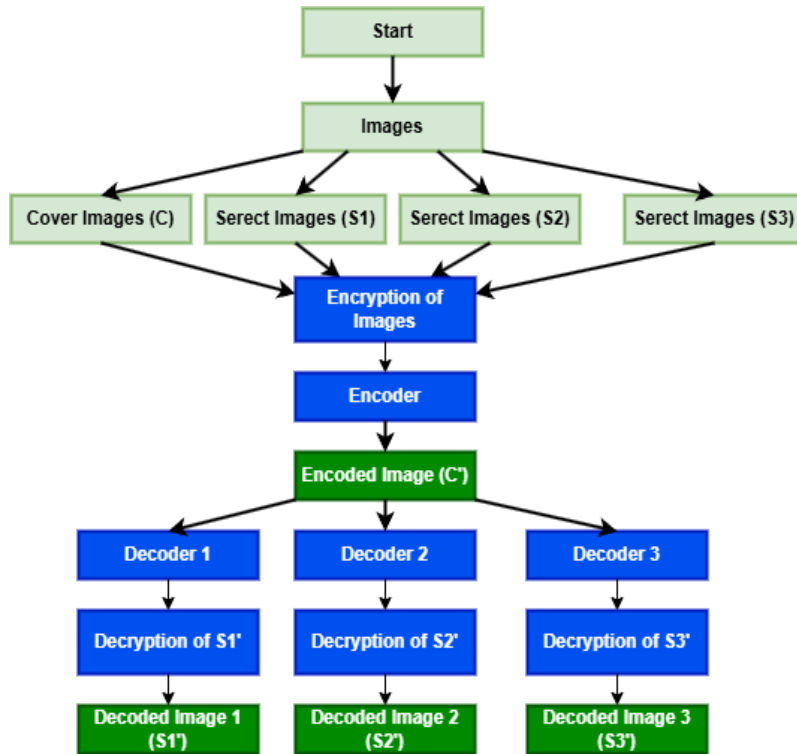


Figure 6.2: Workflow of Our Model with Encryption and Decryption

In our multi-image steganography model, we compared AES and ChaCha20 encryption algorithms using error per pixel and accuracy per pixel metrics. Table 6.1 shows the results numerically, and Figures 6.3 and 6.4 illustrate them visually. For AES, the errors per pixel for S1, S2, S3, and the cover image (C) were 15.66, 17.60, 19.17, and 28.56, respectively, with accuracies of 95.49%, 94.82%, 94.42%, and 91.70%. ChaCha20 showed errors of 18.59 for S1, 18.27 for S2, 16.93 for S3, and 28.01 for C, with accuracies of 94.63%, 94.64%, 95.10%, and 91.82%. Additionally, AES SSIM values were 0.9249 for S1, 0.9415 for S2, 0.9163 for S3, and 0.8821 for C, while ChaCha20 SSIM values were 0.9069 for S1, 0.8886 for S2, 0.9087 for S3, and 0.9224 for C. The PSNR for AES was 26.00 for S1, 26.35 for S2, 26.11 for S3, and 26.27 for C, compared to ChaCha20's PSNR of 24.61 for S1, 27.15 for S2, 23.76 for S3, and 27.11 for C. These results highlight slight differences in performance metrics that could influence the choice of encryption based on specific use cases.

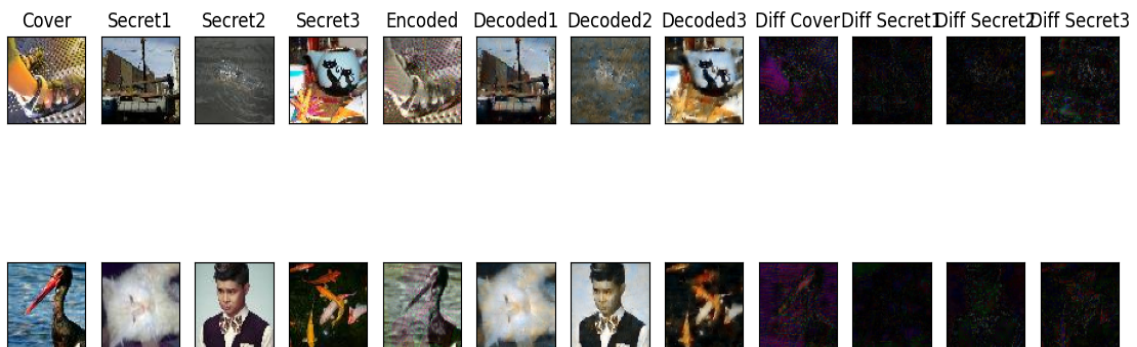


Figure 6.3: Result of Using AES



Table 6.1: Comparison of AES and ChaCha20 Performance Metrics

	<b>S1 Error</b>	<b>S2 Error</b>	<b>S3 Error</b>	<b>C Error</b>
<b>AES</b>	15.661552	17.604502	19.167444	28.555323
<b>ChaCha20</b>	18.585484	18.273357	16.934305	28.00647
	<b>S1 Accuracy</b>	<b>S2 Accuracy</b>	<b>S3 Accuracy</b>	<b>C Accuracy</b>
<b>AES</b>	95.4895%	94.8156%	94.4182%	91.6977%
<b>ChaCha20</b>	94.6343%	94.6384%	95.0981%	91.8197%
	<b>S1 SSIM</b>	<b>S2 SSIM</b>	<b>S3 SSIM</b>	<b>C SSIM</b>
<b>AES</b>	0.924867	0.941468	0.916292	0.882083
<b>ChaCha20</b>	0.9069	0.8886	0.9087	0.9224
	<b>S1 PSNR</b>	<b>S2 PSNR</b>	<b>S3 PSNR</b>	<b>C PSNR</b>
<b>AES</b>	26.0043	26.34675	26.1133	26.2715
<b>ChaCha20</b>	24.6121	27.1450	23.7635	27.1071



Figure 6.4: Result of Using ChaCha20

In our study, we have also implemented the case that is if an attacker got a partial key by writing only the last byte of the key for both AES and ChaCha20. This experiment sought to classify the effectiveness of our encryption in countering such attacks. It emerged that irrespective of the utilization of the modified key, the operation of the decoder yielded errors; in this instance, padding errors with regard to AES. This is because AES and ChaCha20 use the sandboxing process while exact key matching is the decryption step employed by the two ciphers. If one single byte is changed the decryption will not work and the padding will be off if not the whole output might be rendered completely unreadable. This confirms the stability of these algorithms, meaning if one possesses only a part of a key, he will not be able to decrypt the rest parts, thus maintaining the confidentiality and integrity of the concealed images in the steganography context of our model.

## 6.2 Architectural Key Integration for Enhanced Security

In the context of the above-mentioned work, with an aim to introduce the security measures in our multi-image steganography model, we have come up with a new idea of implementing a random key in the architecture of the model. Using this method

also goes beyond image-level security, where encryption is applied to the images and uses the key directly in the encoding and decoding processes. The random key in our model plays a vital role in boosting the security of the encoding and decoding functions by using an architectural level of security. The space of the key is added to the tensor of the combined features of the cover and secret image during the encoding process, therefore resulting in a new transformation of the encoded image with respect to the key introduced. This transformation helps to increase the level of safety of the given image, which is illustrated by the visual embedding of the secret images. During decoding, the same key is subtracted from the tensor in order to gain back the secret images because the reverse process of the above operation is applicable. This architectural approach makes it certain that in the absence of the right key the decoding process will not work and the secret images will remain undecidable. The result of this implementation can be seen in Figure 6.5 and Table 6.2. The results of our multi-image steganography model with architectural key integration are shown in the table below. The error per pixel for secret images S1, S2, and S3 were 38.009476, 34.709866, and 37.02981, respectively, while the cover image (C) had an error of 43.255707. The accuracy per pixel for S1, S2, and S3 were 88.5501%, 89.8921%, and 88.9344%, respectively, and the cover image had an accuracy of 86.8260%. These results demonstrate that incorporating the key directly into the architecture maintains a high level of accuracy and provides robust security by integrating the key into the encoding and decoding processes.

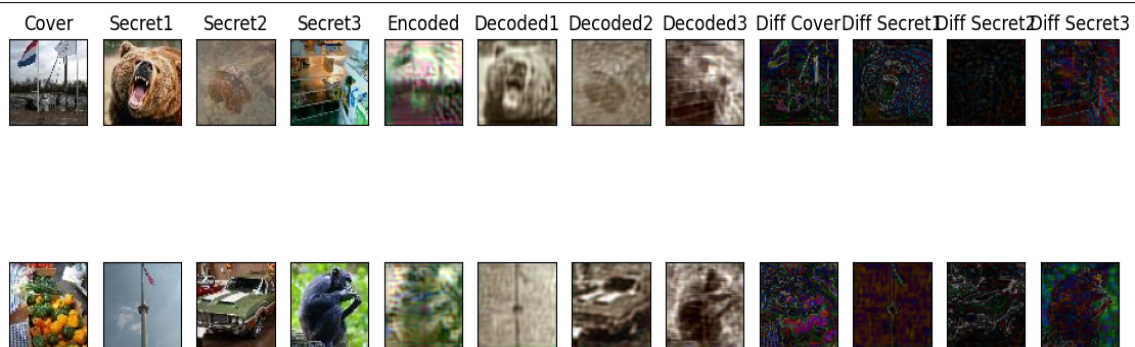


Figure 6.5: Result of Using Architectural Key Integration

<b>S1 Error</b>	<b>S2 Error</b>	<b>S3 Error</b>	<b>C Error</b>
38.009476	34.709866	37.02981	43.255707
<b>S1 Accuracy</b>	<b>S2 Accuracy</b>	<b>S3 Accuracy</b>	<b>C Accuracy</b>
88.5501%	89.8921%	88.9344%	86.8260%

Table 6.2: Architectural Key Integration Performance Metrics

When we introduced an incorrect key into the decoder, the results deteriorated significantly, highlighting the critical importance of using the correct key for decryption in our steganography model. The error per pixel for secret images S1, S2, and S3 increased dramatically to 66.10025, 73.168816, and 51.99589, respectively, with the cover image (C) having an error of 37.371273. Correspondingly, the accuracy per pixel dropped to 79.4211% for S1, 76.5632% for S2, 85.5282% for S3, and



88.2822% for the cover image. These degraded results underscore that the architectural key integration not only secures the encoding and decoding processes but also ensures that any deviation from the correct key renders the extracted images significantly inaccurate and unusable, thereby enhancing the robustness and security of the steganography system against unauthorised access. These results can be seen in Table 6.3 and Figure 6.6

Table 6.3: Performance Metrics with Incorrect Key in Decoder

	<b>S1 Error</b>	<b>S2 Error</b>	<b>S3 Error</b>	<b>C Error</b>
<b>Error per pixel [0, 255]</b>	66.10025	73.168816	51.99589	37.371273
	<b>S1 Accuracy</b>	<b>S2 Accuracy</b>	<b>S3 Accuracy</b>	<b>C Accuracy</b>
<b>Accuracy per pixel [0, 255]</b>	79.4211%	76.5632%	85.5282%	88.2822%

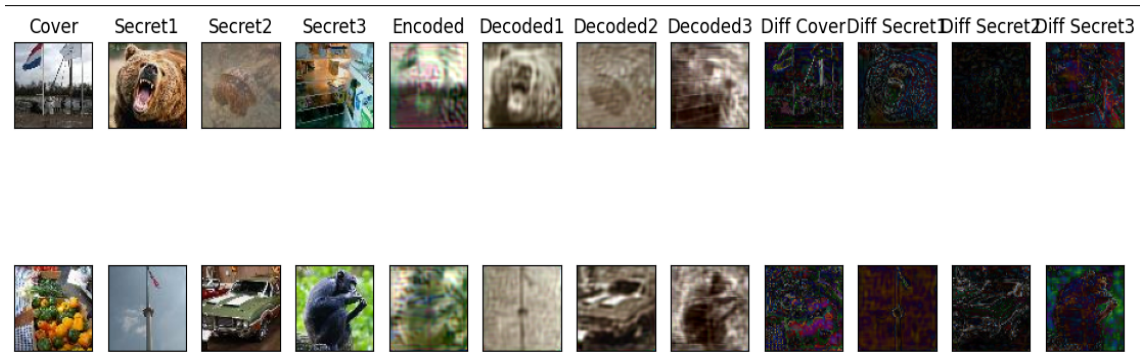


Figure 6.6: Result of Using Architectural Key Integration

Table 6.4: Comparison of AES and ChaCha20 Performance Metrics

	<b>S1 Error</b>	<b>S2 Error</b>	<b>S3 Error</b>	<b>C Error</b>
<b>AES</b>	15.661552	17.604502	19.167444	28.555323
<b>ChaCha20</b>	18.585484	18.273357	16.934305	28.00647
<b>Architectural Key Integration</b>	38.0094	34.7099	37.0298	43.2258
	<b>S1 Accuracy</b>	<b>S2 Accuracy</b>	<b>S3 Accuracy</b>	<b>C Accuracy</b>
<b>AES</b>	95.4895%	94.8156%	94.4182%	91.6977%
<b>ChaCha20</b>	94.6343%	94.6384%	95.0981%	91.8197%
<b>Architectural Key Integration</b>	88.55%	89.89%	88.93%	86.82%
	<b>S1 SSIM</b>	<b>S2 SSIM</b>	<b>S3 SSIM</b>	<b>C SSIM</b>
<b>AES</b>	0.924867	0.941468	0.916292	0.882083
<b>ChaCha20</b>	0.9069	0.8886	0.9087	0.9224
<b>Architectural Key Integration</b>	0.7680	0.7559	0.7879	0.7863
	<b>S1 PSNR</b>	<b>S2 PSNR</b>	<b>S3 PSNR</b>	<b>C PSNR</b>
<b>AES</b>	26.0043	26.34675	26.1133	26.2715
<b>ChaCha20</b>	24.6121	27.1450	23.7635	27.1071
<b>Architectural Key Integration</b>	20.8587	21.9042	20.5512	22.6594

When we attempted to feed an incorrect key into the decoder, we saw the performance decrease drastically, indicating the significance of using the right key for decrypting messages in the steganography model. The number of errors rose significantly, while the levels of accuracy lowered down to a significant measure for all

images. This serves to suggest that the architectural key integration not only safeguards the encoding and decoding processes but also guarantees that any distortion of the correct key makes the extracted images grossly erroneous and useless, all of which can effectively preserve the steganography system from invasion by unauthorised third parties, implying a high level of robustness and security. This is why, with the correct functioning of the key at the architectural level, the model presented a smaller number of errors and higher accuracy, which proves the effectiveness of the presented architectural key solution. AES and ChaCha20 outperform Architectural Key Integration in terms of lower error per pixel, higher accuracy, better SSIM, and higher PSNR, indicating superior image quality and accuracy as seen from table 6.4. AES is preferable for better image quality, while ChaCha20 is suitable for faster processing. This architectural key integration is in its infancy, and less complex than previous studies; however, it has the potential to be applied practically to comparable steganalysis scenarios, providing improved mechanisms for the additional layer of security against duplication attacks in the context of multi-visible image steganography. Contingent research will strive further to describe additional cases of using and developing this methodology.

# Chapter 7

## Conclusion

In conclusion, our steganographic model derived from *CNN* has enhanced the field of multi-image steganography and can be considered as a novelty. The encoder has a multi-scale convolutional architecture which showed outstanding results the decoder also works rather well, which indicates the effectiveness of our model. As expected, the Lrelu activation function displayed superior performance over ReLU and Tanh functions, where lower error per pixel (EPP) and higher accuracy per pixel were achieved. Also, in the architectural key integration; an improvement of the layer of security is achieved as the key is actually incorporated in the encoding and decoding step. This helps to make sure that if the decryption key is not keyed in properly, then decryption will not proceed, hence offering adequate security to the hidden images. The usage of more cryptographic algorithms such as AES as well as ChaCha20 creates a stronger model for security. Our approach can bring in the real-world scenario while keeping up an excellent and extremely secure accuracy. Further development will involve the encoding of more imagery data and increasing the accuracy of the algorithm, as well as the usage of steganalysis methods to improve the anti-protecting method. The future work intends to focus on the enhancement of the model, empowering and increasing its use in the Field of Secure Data and Information Hiding. In the end, out of the three experiments we've conducted, AES yielded better results compared to the other two methods. We assumed that adding a key would enhance security further, but it resulted in lower accuracy. Therefore, future work involves incorporating a key to achieve better results and combining the key with AES encryption.

# Bibliography

- [1] C. M. Bishop, “Neural networks and their applications,” *Review of Scientific Instruments*, vol. 65, no. 6, pp. 1803–1832, Jun. 1994, ISSN: 0034-6748. DOI: 10.1063/1.1144830. eprint: [https://pubs.aip.org/aip/rsi/article-pdf/65/6/1803/8387807/1803\\\_1\\\_online.pdf](https://pubs.aip.org/aip/rsi/article-pdf/65/6/1803/8387807/1803\_1\_online.pdf). [Online]. Available: <https://doi.org/10.1063/1.1144830>.
- [2] T. Morkel, J. H. Eloff, and M. S. Olivier, “An overview of image steganography,” in *ISSA*, vol. 1, 2005, pp. 1–11.
- [3] N. Hamid, A. Yahya, R. B. Ahmad, and O. M. Al-Qershi, “Image steganography techniques: An overview,” *International Journal of Computer Science and Security (IJCSS)*, vol. 6, no. 3, pp. 168–187, 2012.
- [4] S. Baluja, “Hiding images in plain sight: Deep steganography,” in *NIPS*, 2017.
- [5] S. Mukherjee, S. Roy, and P. G. Sanyal, “Image steganography using mid position value technique,” *Procedia Computer Science*, vol. 132, pp. 461–468, Jan. 2018. DOI: 10.1016/j.procs.2018.05.160.
- [6] P. Wu, Y. Yang, and X. Li, “Image-into-image steganography using deep convolutional network: 19th pacific-rim conference on multimedia, hefei, china, september 21-22, 2018, proceedings, part ii,” in Sep. 2018, pp. 792–802, ISBN: 978-3-030-00766-9. DOI: 10.1007/978-3-030-00767-6\_73.
- [7] P. Wu, Y. Yang, and X. Li, “Stegnet: Mega image steganography capacity with deep convolutional network,” *Future Internet*, vol. 10, p. 54, Jun. 2018. DOI: 10.3390/fi10060054.
- [8] A. G. Benedict, “Improved file security system using multiple image steganography,” in *2019 International Conference on Data Science and Communication (IconDSC)*, 2019, pp. 1–5. DOI: 10.1109/IconDSC.2019.8816946.
- [9] X. Duan, K. Jia, B. Li, D. Guo, E. Zhang, and C. Qin, “Reversible image steganography scheme based on a u-net structure,” *IEEE Access*, vol. 7, pp. 9314–9323, 2019. DOI: 10.1109/ACCESS.2019.2891247.
- [10] A. Sajid Ansari, Noida International University, Department of Computer Science and Engineering, NCR Delhi Noida, India, M. Sajid Mohammadi, and M. Tanvir Parvez, “A comparative study of recent steganography techniques for multiple image formats,” *Int. J. Comput. Netw. Inf. Secur.*, vol. 11, no. 1, pp. 11–25, Jan. 2019.
- [11] X. Duan, N. Liu, M. Gou, W. Wang, and C. Qin, “Steganocnn: Image steganography with generalization ability based on convolutional neural network,” *Entropy*, vol. 22, no. 10, p. 1140, 2020. DOI: 10.3390/e22101140.

- [12] X. Duan, L. Nao, G. Mengxiao, *et al.*, “High-capacity image steganography based on improved fc-densenet,” *IEEE Access*, vol. 8, pp. 170 174–170 182, 2020. DOI: 10.1109/ACCESS.2020.3024193.
- [13] I. Kich, “Image steganography by deep cnn auto-encoder networks,” *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 4, pp. 4707–4716, 2020. DOI: 10.30534/ijatcse/2020/75942020.
- [14] S. S. Yadahalli, S. Rege, and R. Sonkusare, “Implementation and analysis of image steganography using least significant bit and discrete wavelet transform techniques,” in *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, 2020, pp. 1325–1330. DOI: 10.1109/ICCES48766.2020.9137887.
- [15] A. Das, J. S. Wahi, M. Anand, and Y. Rana, “Multi-image steganography using deep neural networks,” Jan. 2021. DOI: <https://doi.org/10.48550/arXiv.2101.00350>. arXiv: 2101.00350 [cs.CV].
- [16] S.-P. Lu, R. Wang, T. Zhong, and P. L. Rosin, “Large-capacity image steganography based on invertible neural networks,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 10 811–10 820. DOI: 10.1109/CVPR46437.2021.01067.
- [17] N. Subramanian, O. Elharrouss, S. Al-Maadeed, and A. Bouridane, “Image steganography: A review of the recent advances,” *IEEE Access*, vol. 9, pp. 23 409–23 423, 2021. DOI: 10.1109/ACCESS.2021.3053998.
- [18] A. R. Guzman, *Image steganography using deep learning techniques*, Apr. 2022.
- [19] J. D. Smith and A. B. Johnson, “Understanding neural networks: Structure, training, and applications,” *Journal of Artificial Intelligence*, vol. 17, no. 3, pp. 45–62, 2022.
- [20] B. Wei, X. Duan, and H. Nam, “Image steganography with deep learning networks,” in *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*, 2022, pp. 1371–1374. DOI: 10.1109/ICTC55196.2022.9952432.
- [21] A. Havard, T. Manikas, E. C. Larson, and M. A. Thornton, “CNN-assisted steganography – integrating machine learning with established steganographic techniques,” Apr. 2023. arXiv: 2304.12503 [cs.CR].
- [22] C. Ofoegbu, E. Nwokorie, O. Chinyere, and O. Amadi, “An improved image steganography system using convolutional neural networks,” vol. 4, pp. 1700–1717, Nov. 2023.
- [23] M. Srivastava, P. Dixit, and S. Srivastava, “Data hiding using image steganography,” in *2023 6th International Conference on Information Systems and Computer Networks (ISCON)*, 2023, pp. 1–6. DOI: 10.1109/ISCON57294.2023.10112069.
- [24] <https://www.tibco.com/reference-center/what-is-a-neural-network>, Accessed: 2023-5-19.
- [25] M. Telli, M. Othmani, and H. Ltifi, *An improved image steganography model based on deep convolutional neural networks*, EasyChair Preprint no. 7705, EasyChair, 2022.