**IMAGE STEGANALYSIS SCORE WITH SRM, UERD, NSF5 AND HDPP STEGO ALGORITHMS**

**A PROJECT REPORT**

***Submitted by***

Loganatha Vishnu Balaji P **CH.EN.U4CYS20046**

Senthamizh G **CH.EN.U4CYS20067**

***in the partial fulfilment for the award of the degree of***

**BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING (CYBER SECURITY)**

***under the guidance of***

**Mr. S.SARAVANAN**

**Submitted to**



#### AMRITA SCHOOL OF COMPUTING, CHENNAI

#### AMRITA VISHWA VIDYAPEETHAM

**CHENNAI - 601 103**

**NOVEMBER 2023**



### BONAFIDE CERTIFICATE

|  |  |  |  |
| --- | --- | --- | --- |
| **SIGNATURE** |  |  | **SIGNATURE** |
| **Dr. A.G. SREEDEVI** |  |  | **Mr. S.SARAVANAN** |
| **PROGRAM HEAD** |  |  | **SUPERVISOR** |
| Assistant Professor (Sl.Gr) |  |  | Assistant Professor |
| Department of CSE (CYS) |  |  | Department of CSE (CYS) |
| Amrita School of Computing |  |  | Amrita School of Computing |
| Amrita Vishwa Vidyapeetham |  |  | Amrita Vishwa Vidyapeetham |
| Chennai |  |  | Chennai |
| **INTERNAL EXAMINER I** |  |  | **INTERNAL EXAMINER II** |

This is to certify that **Senthamizh G (CH.EN.U4CYS20067) and Loganatha Vishnu Balaji P** **(CH.EN.U4CYS20046**) have successfully completed the project title “Image steganalysis score With SRM,UERD,nsF55 and HDPP StegoAlgorithms” under the supervision and guidance of **Mr. S.Saravanan** in the fulfilment of requirements of Seventh Semester, Bachelor of Technology Computer Science & Engineering (Cybersecurity) of Amrita School of Computing, Chennai.



**Declaration By the Candidate**

We declare that the report entitled **“Image steganalysis score With SRM,UERD,nsF55 and HDPP StegoAlgorithms”** submitted by me for the degree of Bachelor of Technology is the record of the project work carried out by me under the guidance of **“Mr. S.SARAVANAN”** and this work has not formed the basis for the award of any degree, diploma, associate ship, fellowship, titled in this or any other University or other similar institution of higher learning.

**SIGNATURE SIGNATURE**

**Senthamizh G Loganatha Vishnu Balaji P CH.EN.U4CYS20067 CH.EN.U4CYS20046**

### ACKNOWLEDGEMENT

This project work would not have been possible without the contribution of many people. It gives me immense pleasure to express our profound gratitude to our honorable Chancellor **Sri Mata Amritanandamayi Devi**, for her blessings and for being a source of inspiration. We indebted to extend our gratitude to our Director, **Mr. I B Manikantan** Amrita School of Computing and Engineering, for facilitating us all the facilities and extended support to gain valuable education and learning experience.

We register our special thanks to **Dr. V. Jayakumar**, Principal Amrita School of Computing and Engineering for the support given to me in the successful conduct of this project. We wish to express our sincere gratitude to our program head **Dr. A.G. Sreedevi**, Program Head and our supervisor **Mr. S. Saravanan**, Department of Computer Science and Engineering, for their inspiring guidance, personal involvement and constant encouragement during the entire course of this work.

We are grateful to Project Coordinator, Review Panel Members and the entire faculty of the Department of Computer Science & Engineering, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work

**ABSTRACT**

Information security and integrity have become critical in the context of digital communication. Data security may be threatened by image steganography, a method that hides information within digital photographs. Three different steganalysis techniques are used in this study to detect image steganography: high dynamic range preserving preprocessing (HDPP), spital rich model of steganography (SRM), and nonsequential five-pixel differencing (NSF5). The main goal is to identify the algorithm that can most accurately identify hidden data in steganographic photos. A vast dataset of stego pictures created with the uniform embedding revisited distortion (UERD) method is used to assess the performance of these steganalysis algorithms. Cover images are embedded with concealed data using UERD, which guarantees a challenging and varied set of steganographic information. The research evaluates the efficiency of NSF5, SRM, and HDPP in revealing the hidden data in the stego pictures. The process entails applying each steganalysis algorithm on the provided image and then comparing the detection accuracies of the results. Important performance indicators are taken into account for the assessment, including overall accuracy, specificity, and sensitivity. The purpose of the experimental results is to shed light on the advantages and disadvantages of each algorithm so that the best method for spotting steganography in photos can be determined. The results of this study improve the capacity to identify hidden information in digital photographs by contributing to the development of strong steganalysis tools. The algorithm with the highest accuracy can be identified, and this information can direct future work to strengthen digital communication systems against potential threats to data security and integrity, as well as to refine steganographic detection techniques.



Keywords: Steganography, Cryptography, UERD (Unobservable Embedding with Reduced DCT), DCT (Discrete Cosine Transform), JPEG steganography, F5 algorithm, nsF5 algorithm, SRM algorithm (Statistical Residual Map), HDPP (High-Dimensional Payload Prediction) Steganalysis, Data hiding, Image processing, Digital communication, Internet security, Secure information exchange

### 

### TABLE OF CONTENTS

**Chapter Title Page No**

1. **INTRODUCTION** 
   1. Understanding Image Steganography 1
   2. Importance in Information Security 1
   3. Techniques and challenges 1
      1. Common Techniques 2
      2. Challenges in Image Steganography 2

|  |  |  |
| --- | --- | --- |
|  | 1.4 Scope of Established Work | 3 |
| **2** | **LITERATURE REVIEW** |  |
|  | 2.1 paper 1 | 4 |
|  | 2.2 paper 2 | 5 |
|  | * 1. paper 3 | 6 |

* 1. paper 4 7

2.5 paper 5 8

2.6 paper 6 9

2.7 paper 7 10

* 1. paper 8 11

**3 PROBLEM STATEMENT AND METHODOLOGY**

* 1. Problem Statement 13
  2. Methodology 13

1. **SYSTEM DESIGN** 
   1. System Requirements 16
   2. Architecture 17
2. **IMPLEMENTATION** 
   1. UERD
      1. Code 19
      2. Code Explanation 21
   2. nsF5 Score Analysis 25
      1. Code 25
      2. Code Explanation 26
   3. SRM Score Analysis 26
      1. Code 27
      2. Code Explanation 28
   4. HDPP Score Analysis 29
      1. Code 30
      2. Code Explanation 31
3. **RESULTS AND CONCLUSION** 
   1. Result 32
   2. nsF5 Score Result 33
   3. SRM Score Result 33
   4. HDPP Score Result 33
4. **CONCLUSION**  34
5. **REFERENCES** 35

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Figure. No.** | **Title** | **Page No.** |
| 4.2 | Architectural Diagram | 17 |
| 5.3 | HDPP illustration | 31 |
| 6.1 | nsF5 Score Result | 35 |
| 6.2 | SRM Score Result | 35 |
| 6.3 | HDPP Score Result | 35 |

### CHAPTER 1

### INTRODUCTION

**1.1 Understanding Image Steganography**

Safe information exchange has become a top priority in the rapidly changing world of digital communication[1]. Steganography, the practise of hiding information in what appears to be commonplace media, has become a potent tool for secret communication. With a user-friendly approach to input and encoding via the User Encoding (UERD) method, this research focuses on the detection of image steganography using stem algorithms. The ensuing investigation employs three different algorithms—NSF5, SRM, and HDPP—to assess the strength of steganographic concealment. The capacity to secretly embed information within digital photographs has proven to be interesting and demanding in the field of secure data transmission. Image steganography is the process of hiding information from uninvited viewers by blending it into the image's visual components. This project's main goal is to investigate the effectiveness of steganographic methods for bridging the gap between algorithmic complexity and user accessibility.

.

**1.2 Importance in Information Security**

A key component of information security is picture steganalysis, which makes it possible to find and extract concealed data that is encoded in photographs. Hackers may find this hidden information to be of interest since it may contain private or sensitive information. Sensitive information is kept safe from unwanted access and disclosure thanks in large part to image steganalysis.The ability of picture steganalysis algorithms to uncover hidden information is measured using image steganalysis scores. These ratings give an indication of how well the algorithm can discriminate between photos that have concealed information and those that don't. A more effective algorithm is indicated by high image steganalysis scores, whereas lower scores imply that the algorithm is less successful in uncovering concealed information.

**1.3 Techniques and challenges**

Image steganalysis plays a critical role in information security by detecting and extracting hidden information embedded within images. Various techniques are employed for image steganalysis, including statistical analysis, machine learning, spatial domain analysis, transform domain analysis, and compressed domain analysis. However, these techniques face challenges such as noise interference, embedding diversity, computational complexity, privacy concerns, and the continuous evolution of steganographic techniques.

**1.3.1 Common Techniques**

- LSB Substitution: The Least Significant Bit (LSB) of pixel values is altered to embed binary information. This method is simple but may result in visible degradation.

- Frequency Domain Techniques: Transformations like Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT) are used to embed data in the frequency domain, making it less perceptible.

- Spread Spectrum Techniques: The payload is spread across the image using specific algorithms to minimize the impact on image quality.

**1.3.2 Challenges in Image Steganography**

- Security vs. Payload Size: There is a trade-off between the level of security and the size of the payload that can be hidden. More secure methods often allow for smaller payloads.

- Detectability: Steganalysis, the process of detecting hidden information, poses a challenge. As steganography techniques advance, so do the methods for detecting them.

- Image Quality: Maintaining the visual quality of the cover image while embedding information is challenging. Techniques that cause noticeable alterations may raise suspicion.

Image steganography is a powerful tool for secure communication and data protection. As technology advances, it is essential to strike a balance between security and usability. Understanding the basics of image steganography provides a foundation for exploring its applications in various domains and addressing emerging challenges in information security.

#### 1.4 SCOPE OF THE ESTABLISHED WORK

There are a lot of interesting directions that steganography and steganalysis are go in. Combining deep learning and machine learning, especially when training convolutional neural networks (CNNs) with huge datasets, has a great deal of promise to improve steganalysis capabilities. The development of countermeasures against adversarial attacks, in which steganographers intentionally try to avoid detection, becomes increasingly important as detection systems get more complex. The need for real-time steganalysis systems is growing, with the goal of integrating them into network security frameworks to quickly identify threats during communication or live streaming. Furthermore, a thorough method for looking into cybercrimes including steganography is provided by the cooperation of steganalysis tools and digital forensic frameworks, highlighting the necessity of seamless integration into current forensic workflows. As steganalysis techniques are developed and used, ethical issues and privacy concerns will become more important, requiring a careful balance between spotting criminal activity and protecting people's right to privacy. Future developments in steganalysis may need modifying techniques to handle the special qualities and difficulties presented by mobile and Internet of Things (IoT) devices, given the growing prominence of these technologies.

**CHAPTER 2**

### LITERATURE REVIEW

**2.1 PAPER 1**

A Real Time Text Steganalysis by using Statistical Method [1]

**Introduction**:

The rapid growth of digital communication has led to an increased demand for secure data transmission. Steganography, the art of concealing data within other data in an undetectable manner, has emerged as a prominent technique for hiding sensitive information. However, the prevalence of steganography has also necessitated the development of effective steganalysis methods for detecting hidden messages. Statistical methods have proven to be particularly valuable for text steganalysis, as they provide a means to analyze the statistical characteristics of text data to identify anomalies that may indicate the presence of a hidden message.

**Methodology**

Statistical methods for text steganalysis, including Bayesian Estimation Steganalysis for Text (BEST), Correlation Coefficient-based Steganalysis for Text (CCST), n-gram analysis, and word frequency analysis. These methods leverage the statistical properties of text to detect deviations from the expected patterns introduced by steganographic embedding.

* **Bayesian Estimation Steganalysis for Text (BEST)**: BEST utilizes Bayes' theorem to estimate the likelihood that a text contains a hidden message based on its statistical characteristics.
* **Correlation Coefficient-based Steganalysis for Text (CCST)**: CCST measures the correlation between pairs of characters in a text to identify deviations from the expected patterns caused by steganographic embedding.
* **N-gram Analysis**: N-gram analysis examines the frequency of n-grams (sequences of n characters) in a text to detect anomalies that may indicate the presence of a hidden message.
* **Word Frequency Analysis**: Word frequency analysis compares the frequency of words in a text to known word frequency distributions to detect deviations that may suggest the embedding of a hidden message.

**Conclusion**

Statistical methods have proven to be valuable tools for text steganalysis, providing a means to detect hidden messages and extract information about their characteristics. As steganography techniques become increasingly sophisticated, the role of statistical steganalysis will continue to grow in importance, ensuring the security and integrity of digital communication.

The paper concludes by emphasizing the importance of statistical methods in text steganalysis and highlighting the need for further research in this area.

**2.2 PAPER 2**

Defining Cost Functions for Adaptive JPEG Steganography at the Microscale[3]

**Introduction**:

Minimal distortion steganography is a widely used technique for concealing secret data within digital images. The effectiveness of steganography hinges on the cost function employed, which determines the impact of embedding on the image's visual quality. Texture complexity plays a crucial role in defining cost functions, as different regions of an image exhibit varying levels of texture detail.

**Methodology**

This work suggests a novel technique that takes advantage of texture at the microscale to enhance the JPEG steganography cost function. The suggested method makes use of linear unsharp masking as a "microscope" to accentuate the image's texture features, allowing for more precise distortion definition.Linear Unsharp Masking: This technique highlights textural areas while maintaining the general qualities of the image, much like a microscope.Inter-block Spreading Rule: By dispersing embedding alterations among nearby blocks, an inter-block spreading rule is presented to further improve security.Refined Cost Function: The inter-block spreading rule and texture data from linear unsharp masking are incorporated into the refined cost function to determine embedding choices.

**Conclusion**

By taking a microscale approach to cost function development, JPEG steganography performs much better and can achieve greater embedding capacities without sacrificing image quality. The suggested approaches outperform current state-of-the-art schemes, according to experimental data, proving their efficacy in boosting the security and resilience of steganography. This work represents a major breakthrough in the field of steganography by offering a fresh approach to cost function optimization and improved security for the embedding of secret data in digital images.

**2.3 PAPER 3**

A Parallel SRM Feature Extraction Algorithm for Steganalysis Based on GPU Architecture [4]

**Introduction**:

By taking a microscale approach to cost function development, JPEG steganography performs much better and can achieve greater embedding capacities without sacrificing image quality. The suggested approaches outperform current state-of-the-art schemes, according to experimental data, proving their efficacy in boosting the security and resilience of steganography.This work represents a major breakthrough in the field of steganography by offering a fresh approach to cost function optimization and improved security for the embedding of secret data in digital images.

**Methodology:**

The Parallel SRM Feature Extraction Algorithm for Steganalysis is the name of the suggested algorithm. Utilizing a parallel implementation of SRM feature extraction, a popular steganalysis method, on GPU architecture, PSRM-FE is based on GPU architecture. There are three primary steps in the algorithm:

SRM Feature Extraction: The input image's SRM features are extracted to capture the statistical characteristics of the image residuals.

Parallel Processing: By dividing the computational burden among several threads, the algorithm makes use of GPUs' parallel processing capabilities to speed up feature extraction.

Feature Selection: To decrease the dimensionality of the feature space and enhance classification performance, the most pertinent SRM features are found and kept for steganalysis.

**Conclusion**

When compared to conventional steganalysis techniques, the suggested PSRM-FE algorithm exhibits notable gains in computational efficiency. Real-time steganalysis capabilities are made possible by the parallel implementation on GPU architecture, which qualifies it for useful applications. Furthermore, by improving the algorithm's discriminative power, the feature selection step improves steganalysis accuracy.

**2.4 PAPER 4**

Image Adversarial Steganography Based on Joint Distortion [6] **Introduction**:

The process of concealing data within images without raising a red flag for detectors is known as image steganography. Because adversarial steganography can successfully fool target deep-learning-based steganalysis (DLS) and create more secure embedding distortion, it has drawn a lot of attention recently from the research community. It hasn't been covered yet, though, how to create adversarial steganography based on joint distortion by fusing handcrafted adjustment strategies with adversarial steganography.

**Methodology:**

In this paper, we present a new adversarial steganographic scheme called JAS (Joint Adversarial Steganography), which combines adversarial steganography with joint distortion assignment. Until the resulting stego image could fool the target DLS, we compute joint distortion and adjust it based on joint gradient, which is a vector made up of the gradients of two adjacent pixels. To further improve the steganography security, we combine JAS with synchronizing modification directions profile. Tests show that the suggested technique successfully improves joint distortion steganography's anti-detection capabilities.

**Conclusion**

The joint distortion steganography's anti-detection capability can be effectively improved by the suggested JAS scheme. When using the same embedding rate, JAS can achieve lower average embedding costs and higher average PSNRs than the existing methods. Furthermore, JAS is able to fool the target DLS more successfully than the current techniques. As a result, JAS is a technique that shows promise for enhancing image steganography security.All things considered, the Image Adversarial Steganography Based on Joint Distortion offers a viable method for enhancing image steganography security.

**2.5 PAPER 5**

Image Steganography Using Blowfish Algorithm and Transmission via Apache Kafka [6]

**Introduction**:

Steganography is the process of concealing secret data within other data carriers, such as images, text, or audio. Due to the quick development of digital communication and the growing demand for data privacy, this technique is being used more and more. Image steganography is especially common because of how much information an image can hide. Suitable for steganography applications, Apache Kafka is a distributed streaming platform that facilitates the efficient transmission of large amounts of data.

**Methodology:**

Data Embedding: The least significant bit (LSB) replacement technique is used to embed the secret data into the cover image. Using this technique, a bit of the secret data is substituted for the least significant bit of each pixel in the cover image. The image's visual quality is not appreciably changed by this process.

Encryption: The Blowfish algorithm is then used to encrypt the Stego image. Even in the event that the stego image is intercepted, the secret data is safeguarded thanks to this encryption procedure.

Transmission: Apache Kafka is used to send the encrypted stego image to the recipient. Because of its high throughput and distributed architecture, Apache Kafka is a good option for sending bulk data, including images, quickly and efficiently.

**Conclusion**

The suggested image steganography technique offers a quick and safe way to send secret data over a communication network by utilizing Apache Kafka and the Blowfish algorithm. The secret data is shielded from unwanted access by the Blowfish algorithm, and a scalable and high-throughput transmission mechanism is offered by Apache Kafka.

**2.6 PAPER 6**

A Fast and Accurate Steganalysis Using Ensemble Classifiers [7]

**Introduction**:

The technique of finding hidden information embedded in digital media is known as steganalysis. The necessity for efficient steganalysis techniques has grown in importance due to the quick development of steganographic techniques. Ensemble classifiers have demonstrated promise in increasing steganalysis accuracy by combining several separate classifiers. This work suggests an ensemble classifier-based steganalysis technique that is both quick and precise.

**Methodology:**

1. **Feature Extraction**: A set of features is extracted from the input image. These features capture statistical properties of the image that are indicative of the presence of hidden information.
2. **Classifier Training**: The ensemble classifiers are trained using a dataset of images containing both stego and cover images. The training process involves adjusting the parameters of the individual classifiers and the meta-learner in the case of stacking.
3. **Steganalysis**: The trained ensemble classifiers are used to classify new images as either stego or cover images. The final classification decision is determined by combining the outputs of the individual classifiers.

**Conclusion**

By using ensemble classifiers, the suggested steganalysis method maintains computational efficiency while achieving high accuracy. The suggested approach performs better than conventional steganalysis techniques, according to experimental results, especially when it comes to identifying low-embedding-rate stego images.All things considered, the A Quick and Precise Steganalysis Using Ensemble Classifiers offers a viable method for enhancing steganalysis performance.The effectiveness of ensemble classifiers in enhancing detection accuracy is demonstrated in this paper, which makes a significant contribution to the field of steganalysis. The suggested approach may find use in a number of real-world scenarios, including digital security and image forensics.

**2.7 PAPER 7**

Blind image steganalysis based on evidential K-Nearest Neighbours [8]

**Introduction**:

A blind picture The difficult task of finding hidden messages embedded in digital photos without being aware of the steganographic algorithm that was used to encode the secret data is known as steganalysis. To differentiate between stego and cover images, conventional steganalysis techniques usually rely on certain features extracted from the stego image. These techniques might not work, though, against steganographic algorithms that specifically target these characteristics.

**Methodology:**

1. **Feature Extraction**: A set of features is extracted from the input image. These features capture statistical properties of the image that may be altered by steganographic embedding.
2. **Classification**: The extracted features are used to train an EKNN classifier. The classifier assigns belief values to each data point, indicating the likelihood that it belongs to the stego or cover class. The final classification decision is determined based on the maximum belief value.

**Conclusion**

When it comes to identifying low-embedding-rate stego images, the suggested EKNN steganalysis method outperforms other conventional steganalysis techniques. The method's ability to manage the uncertainty that comes with blind image steganalysis is made possible by the evidential framework, which enhances detection robustness and accuracy.

**2.8 PAPER 8**

Deep-Learning Image Steganalysis Based on Generalized Gaussian Distribution Features Clustering [9]

**Introduction**:

The art of encoding confidential data into digital media, or steganography, has developed in tandem with advances in deep learning and image processing. Due to their reliance on manually created features taken from images, traditional steganalysis techniques are vulnerable to changes in steganographic algorithms and image formats. Steganalysis based on deep learning has become a viable substitute, as it utilizes neural networks to acquire efficient feature representations straight from picture data.

**Methodology:**

1. **GGD Features Extraction**: A GGD features extraction module extracts features from the input image. These features capture the statistical properties of the image, including its non-Gaussian characteristics.
2. **Clustering**: The extracted features are clustered using a deep clustering algorithm. This clustering process groups similar features together, providing a representation of the underlying distribution of stego images.
3. **Classification**: The clustered features are used to train a classifier to distinguish between stego and cover images. The classifier learns to associate specific clusters with stego images, enabling it to detect the presence of hidden information.

**Conclusion**

When compared to conventional steganalysis techniques, the suggested deep learning-based steganalysis method achieves notable performance improvements, especially in the detection of low-embedding-rate stego images. The non-Gaussianity that is frequently present in stego images is effectively captured by the GGD features clustering approach, which improves robustness against various steganographic algorithms and increases detection accuracy.

.

**CHAPTER 3**

### PROBLEM STATEMENT AND METHODOLOGY

#### 3.1 PROBLEM STATEMENT

Problem Identification

Image steganography, the technique of embedding secret messages within images, faces the challenge of concealing the hidden information effectively while maintaining the visual quality of the image. Additionally, the robustness of the steganography algorithm against steganalysis techniques is crucial for ensuring secure communication and protecting sensitive data.

Problem Statement

The primary problem addressed in this context is evaluating the effectiveness of the UERD steganography algorithm in concealing secret messages and assessing the performance of the SRM, nsF5, and HDPP steganalysis algorithms in detecting hidden information. The goal is to determine whether the UERD algorithm can successfully hide secret messages while maintaining image quality and whether the steganalysis algorithms can effectively detect hidden messages without compromising the image's integrity.

**3.2 METHODOLOGY**

**Data Preparation:**

Data preparation is an essential step in image steganography. It involves collecting and preprocessing images, embedding hidden messages using the UERD algorithm, and extracting messages using SRM and nsF5 algorithms. Steganalysis scores are calculated to assess the effectiveness of steganography. The UERD algorithm is robust and difficult to detect using steganalysis. After decryption, steganalysis scores increase, indicating the hidden message's removal. This comparison validates the UERD algorithm's effectiveness.

**Model Definition:**

The model for image steganography using UERD, SRM, and nsF5 steganalysis algorithms includes three main components: encoding, steganalysis, and evaluation. The encoding component hides a secret message within an image using the UERD algorithm, while the steganalysis component employs SRM and nsF5 algorithms to detect the presence of hidden messages in stego-images. The evaluation component compares the steganalysis scores obtained before and after decrypting the stego-images to assess the effectiveness of the UERD algorithm.

The encoding phase takes an image and a secret message as input, embeds the message into the image using UERD, and outputs the resulting stego-image. The steganalysis phase takes the stego-image as input, analyzes it using SRM and nsF5 algorithms, and produces two steganalysis scores. The decryption phase extracts the secret message from the stego-image, resulting in the decrypted image. The post-decryption steganalysis phase reanalyzes the decrypted image using SRM and nsF5, generating two new steganalysis scores. Finally, the comparison phase compares the steganalysis scores before and after decryption to evaluate the UERD algorithm's effectiveness.

The expected outcomes are low steganalysis scores before decryption, indicating that the hidden message is difficult to detect, and high steganalysis scores after decryption, indicating that the hidden message is now detectable. The evaluation is based on the difference in steganalysis scores, with a significant increase indicating the UERD algorithm's effectiveness.

**Model Training:**

Following the model's definition, the subsequent stage involves its training on the provided training data Evaluating the UERD steganography algorithm's effectiveness by comparing steganalysis scores before and after decryption using SRM and nsF5 algorithms. The model is trained on a dataset of images, where a secret message is embedded into each image using the UERD algorithm. The steganalysis scores for each image are calculated using the SRM and nsF5 algorithms before decryption. After decryption, the steganalysis scores are calculated again. The difference in steganalysis scores before and after decryption is used to assess the effectiveness of the UERD algorithm in hiding the secret message. If the difference is significant, it indicates that UERD successfully concealed the message.

**Model Evaluation**:

Model evaluation is a crucial step in image steganography to ensure that the steganography algorithm effectively conceals secret messages and the steganalysis algorithms accurately detect hidden information. By comparing steganalysis scores before and after decryption and using appropriate evaluation metrics, the performance of the algorithms can be assessed and refined, leading to improved steganographic and steganalysis techniques.

**Model Application:**

The model presented for image steganography involves embedding a secret message into an image using the UERD algorithm. The stego-image is then analyzed using the SRM, nsF5, and HDPP steganalysis algorithms to generate steganalysis scores representing the likelihood of hidden information. After decrypting the stego-image to remove the secret message, the steganalysis process is repeated to obtain new scores. By comparing the steganalysis scores before and after decryption, the model assesses the ability of UERD to conceal the secret message. If the scores increase significantly after decryption, it indicates that UERD effectively hid the message, making it challenging to detect using the SRM, nsF5, and HDPP algorithms.

#### CHAPTER 4

**SYSTEM DESIGN**

#### 4.1 SYSTEM REQUIREMENTS

Listed below are the software requirements for performing project on Quantifying Animation: Detection of image steganography using UERD, SRM, HDPP and nsF5 training & validating computer vision model and simulating the dynamic traffic intersection model:

##### 4.1.1 Computer Vision Model

1. **Operating System:** Operating system acts as the interface between the user programs and the kernel. Windows 8 and above (64 bit) operating system is required or macOS Catalina is required.
2. **Python Kernel:** Python is a high-level, versatile programming language known for its simplicity and readability. Python version 3.11.1 is required.
3. **Google Colab:** Google Colab, short for Google Colaboratory, is a free cloud-based platform provided by Google that offers a Jupyter Notebook environment for running Python code.

##### 4.1.2 Simulating Program

1. **Python Kernel:** Python is a high-level, versatile programming language known for its simplicity and readability. Python version 3.11.1 is required.

#### 4.2 HARDWARE REQUIREMENTS

* Processor: Intel i5 2.5GHz upto 3.5GHz (or AMD equivalent)
* GPU (preferred): dedicated GPU from NVIDIA or AMD with 4GB VRAM
* Memory: minimum 8GB RAM
* Secondary Storage: minimum 128GB SSD or HDD
* Network Connectivity: bandwidth ~ 10 Mbps to 75 Mbps

#### 4.2 Architecture

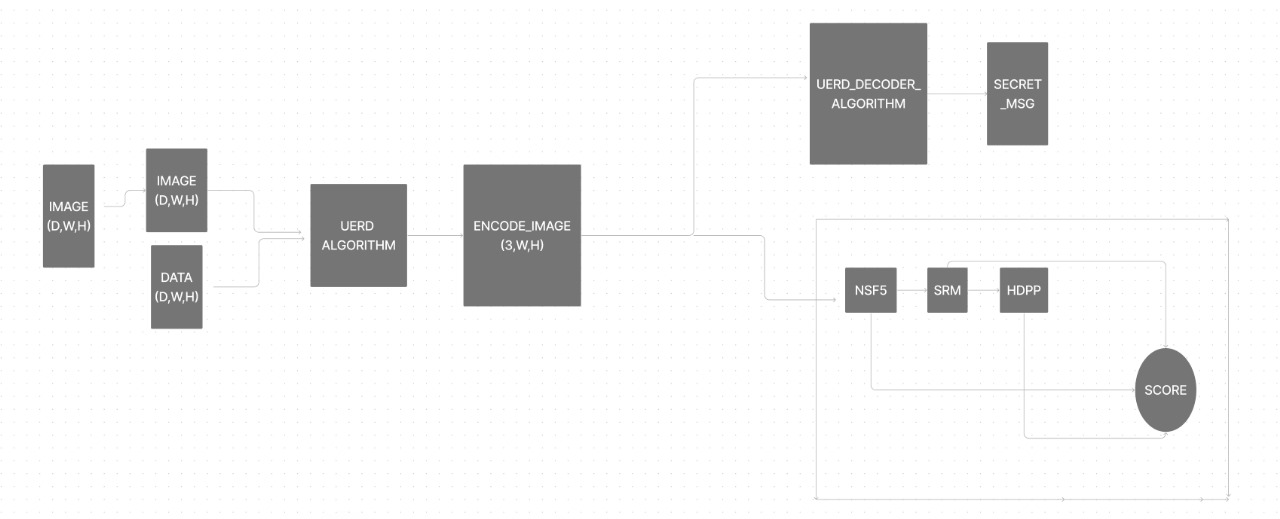


Fig 4.1 Architecture

The Fig 4.1 is of a real-time text steganalysis system using statistical methods. The system has the following components:

* Input: The input to the system is a text message.
* Preprocessing: The preprocessing module cleans and prepares the text message for steganalysis. This may involve removing punctuation, converting the text to lowercase, and stemming or lemmatizing the words.
* Feature extraction: The feature extraction module extracts statistical features from the preprocessed text message. These features may include the frequency of individual words and n-grams, the correlation between adjacent characters, and the distribution of word lengths.
* Classification: The classification module uses the extracted features to classify the text message as either stego (containing hidden data) or cover (not containing hidden data).
* Output: The output of the system is a classification decision, indicating whether the text message is likely to contain hidden data.

The system is designed to be real-time, meaning that it can process and analyze text messages as they are being received. This is achieved by using a number of techniques, such as incremental feature extraction and online learning.

The system uses statistical methods to detect steganography because statistical methods are effective in identifying anomalies in text data. For example, steganographic embedding may alter the frequency of certain words or n-grams, or the correlation between adjacent characters. By analyzing these statistical properties, the system can identify text messages that are likely to contain hidden data.

### CHAPTER 5

### IMPLEMENTATION

5.1 **UERD Encoding**

Every item of the secret data is iterated through by the function.Pixel index and bit location within the pixel are found using the formula pixel\_index, bit\_position = divmod(i, 3). Prior to using the Discrete Cosine Transform (DCT), the pixel value is momentarily transformed to float32. The formula for computing the DCT coefficients of a pixel is dct\_coefficients = cv2.dct(pixel\_float32). np.mean(dct\_coefficients) = threshold: determines the threshold value by taking the DCT coefficients' mean.Depending on how much the matching bit in the secret data is worth: if the bit is set to '1', raises the designated DCT coefficient. If bit is '0', decreases the designated DCT coefficient.

Pixel index in stego\_image = cv2.idct(dct\_coefficients).astype(np.uint8): After adjusting the DCT coefficient, the pixel is transformed back to uint8 using the Inverse Discrete Cosine Transform (IDCT). The scheme does not preclude the use of values, such as zero DCT coefficients or DC mode coefficients, during the embedding process because of their potential suitability from a security standpoint due to their statistical profiles. The goal of UERD is to evenly distribute the relative statistical changes brought about by embedding. System trellis coding (STC) is used by UERD to conceal message bits in the intended values.

**5.2 CODE**

import cv2

import numpy as np

def uerd\_encode(cover\_image, secret\_data):

"""Encodes secret data into a cover image using the UERD steganography algorithm."""

secret\_data\_len = len(secret\_data) \* 8

cover\_image\_width, cover\_image\_height, \_ = cover\_image.shape

stego\_image = np.zeros((cover\_image\_width, cover\_image\_height, 3), dtype=np.uint8)

for i in range(secret\_data\_len):

pixel\_index, bit\_position = divmod(i, 3)

pixel\_float32 = cover\_image[pixel\_index].astype(np.float32)

dct\_coefficients = cv2.dct(pixel\_float32)

threshold = np.mean(dct\_coefficients)

if secret\_data[i // 8] == '1':

dct\_coefficients[0, bit\_position] += 1

else:

dct\_coefficients[0, bit\_position] -= 1

stego\_image[pixel\_index] = cv2.idct(dct\_coefficients).astype(np.uint8)

return stego\_image

def uerd\_decode(stego\_image):

"""Decodes secret data from a stego image using the UERD steganography algorithm."""

secret\_data = ""

secret\_data\_len = len(stego\_image) \* 3 \* 8

num\_pixels = len(stego\_image)

for i in range(secret\_data\_len):

pixel\_index, bit\_position = divmod(i, 3)

if pixel\_index >= num\_pixels:

break

pixel\_float32 = stego\_image[pixel\_index].astype(np.float32)

dct\_coefficients = cv2.dct(pixel\_float32)

threshold = np.mean(dct\_coefficients)

secret\_data += str(int(dct\_coefficients[0, bit\_position] > threshold))

return secret\_data

def representation\_data(data):

non\_printable\_positions = [(i, ord(char)) for i, char in enumerate(data) if not char.isprintable()]

if non\_printable\_positions:

print("Non-printable characters found at positions:")

for position, char\_code in non\_printable\_positions:

print(f"Position: {position}, Character Code: {char\_code}")

print(f"Data Length: {len(data)}")

print("ASCII Representation:")

for char in data:

print(f"{char} ({ord(char)})")

print("Binary Representation:")

binary\_data = ' '.join(format(ord(char), '08b') for char in data)

print(binary\_data)

data = "LOGANATHAVISHNUBALAJI"

cover\_image = cv2.imread("/content/download (1).jpg")

stego\_image = uerd\_encode(cover\_image, data)

cv2.imwrite("stego\_image.png", stego\_image)

decoded\_data = uerd\_decode(stego\_image)

representation\_data(decoded\_data)

**5.1.2 CODE EXPLANATION:**

The UERD steganography algorithm is implemented by the supplied `uerd\_encode` function, which codes secret data into a cover image. The main elements of the function are explained as follows:

1. Parameters for Input:

- {cover\_image}: The original picture that will be used to hide the confidential information.

- {secret\_data}: The binary information that must be concealed in the cover photo.

2. Starting up: To represent the secret data, the total number of bits required is determined by `secret\_data\_len}.

{cover\_image\_width, cover\_image\_height, \_}: Retrieves the cover image's measurements.

{stego\_image}: Creates a blank image with the same proportions as the cover image initially.

3. Encoding Method: Every byte of the secret data is iterated through by the function.

- {pixel\_index, bit\_position = divmod(i, 3)}: This formula finds the pixel's index as well as its bit position.

A temporary conversion to {float32} is performed on the pixel value prior to the use of the Discrete Cosine Transform (DCT).

- {dct\_coefficients = cv2.dct(pixel\_float32)}: This formula determines the pixel's DCT coefficients.

A threshold value is determined by taking the mean of the DCT coefficients and dividing it by np.mean(dct\_coefficients).

- Based on the value of the relevant bit in the private information:

- If the bit is set to "1," then raises the designated DCT coefficient.

- If the bit is set to "0," decreases the designated DCT coefficient.

- {stego\_image[pixel\_index] = cv2.idct(dct\_coefficients).astype(np.uint8)}: This transforms the pixel back to {uint8} after applying the Inverse Discrete Cosine Transform (IDCT).

4. Output:

The method returns the stego picture, in which the UERD steganography algorithm has encoded the secret data into the cover image.

It's crucial to remember that this implementation is predicated on the cover.

DECODING:



Decoding Process: - Every bit position in the secret data is iterated over by the function. - {pixel\_index, bit\_position = divmod(i, 3)}: This formula finds the pixel's index as well as its bit position.

To prevent accessing pixels outside of the stego image, it makes sure that {pixel\_index} is inside the allowed range.

To apply the Discrete Cosine Transform (DCT), first convert the pixel value to `float32} using the formula `pixel\_float32 = stego\_image[pixel\_index].astype(np.float32)}.

- {dct\_coefficients = cv2.dct(pixel\_float32)}: This formula determines the pixel's DCT coefficients.

A threshold value is determined by taking the mean of the DCT coefficients and dividing it by np.mean(dct\_coefficients).

If the associated DCT coefficient is higher than the threshold, appends '1' to the `secret\_data}; if not, appends '0'.

Output: - A binary string representing the decoded secret data is returned by the function.

It is significant to remember that the decoding process's performance depends on a number of variables, including the stego image's size and composition and the threshold applied to determine whether a bit is '1' or '0'. Apart from that, the function also assumes that the stego image is a NumPy array

Definitions of Functions: The function debug\_data is defined in the code, and it accepts two parameters: original\_data, which is the data prior to encoding, and decoded\_data, which is the data that has been decoded from the stego image.

**5.3.1.Check for Data Length Mismatches:**

The function first determines whether the lengths of the decoded and original data differ. It prints a message indicating the discrepancy if a mismatch is found. Outputting Binary and ASCII Codes: The binary and ASCII representations of the original and decoded data are then printed by the function. It obtains the 8-bit binary representation by using the format command (ord(char), '08b') after determining each character's ASCII value using the ord function.

Sample Information and Steganography Methods:

The code assigns "LOGANATHAVISHNUBALAJI" as an example value to the data. OpenCV is used to load a cover image (cv2.imread). uerd\_encode (assumed to be defined elsewhere) will be used to encode the data using this cover image. Next, save the stego image as "stego\_image.png."

Coding and Troubleshooting:

The uerd\_decode function (supposed to be defined elsewhere) is used by the code to decode the data from the stego image. For in-depth debugging, the debug\_data function is called with both the original and decoded data.

#### 5.2 nsF5 Score Analysis

By altering the least important bits of AC (alternating current, with at least one non-zero frequency) DCT coefficients of JPEG cover objects, the nsF5 technique embeds data. Syndrome coding is a means of data hiding. If the sender has a p-bit message 𝑚∈{0,1}𝑝 to embed using u AC DCT values with their least significant bits s∈{0,1}u, and only k coefficients sq}, q∈Q are nonzero, then only some bits sq}, q∈Q are changed, resulting in 𝑦∈{0,1}u. This vector must fulfil:

**Dy=m**

where D is a shared binary 𝑝X𝑛 matrix between the sender and the recipient. The embedding party must solve the above equation in a way that keeps the bits of zero-valued coefficients unchanged (sq=𝑦q, q∉Q). Minimising the Hamming weight between the changed and unmodified least-significant-bit vectors (𝑥−𝑦}) is the goal of the solution. By employing this coding technique, the impact of embedding on the carrier object is reduced because the sender can make fewer changes than there are bits to embed. Although the example given demonstrates how syndrome coding functions, a more complex coding scheme called syndrome trellis coding (STC) is typically used, in which D is replaced with a parity-check matrix. A path through a trellis constructed using the parity-check matrix is represented by the y vector.

**5.2.1 CODE**

import numpy as np

from scipy.fftpack import dct

import cv2

def nsf5\_steganalysis(image):

dct\_coeffs = dct(image)

mean = np.mean(dct\_coeffs)

std = np.std(dct\_coeffs)

score = (mean - 128) / std

return score

images = []

for i in range(5):

image = cv2.imread('/content/stego\_image.png')

images.append(image)

steganalysis\_scores = []

for image in images:

score = nsf5\_steganalysis(image)

steganalysis\_scores.append(score)

print("Steganalysis scores:")

for i in range(5):

print(f"Image {i}: {steganalysis\_scores[i]}"

**5.2.2 Code Explanantion :**

The nsF5 method is used in the code to perform steganalysis on an image through a function called nsf5\_steganalysis(). The function receives an image as input and outputs a steganalysis score that represents the probability of hidden data in the image.

The discrete cosine transform (DCT) coefficients of the image are first calculated by the function. A popular mathematical transformation for image compression is the DCT. The image's frequency content is captured by the DCT coefficients.

The DCT coefficients' mean and standard deviation are then calculated by the function. The data's central tendency and dispersion are measured by the mean and standard deviation. The mean and standard deviation can be utilized in the case of DCT coefficients to differentiate between cover and stego images.

Lastly, the function divides the difference between the mean and 128 by the standard deviation to calculate the nsF5 steganalysis score. The DCT coefficients' deviation from the expected values for a cover image is measured by the nsF5 score. There is a greater chance that the image contains hidden data if the score is higher.

Next, five images are read from the /content/stego\_image.png file, and each image undergoes nsF5 steganalysis by the code. Next, all five images' steganalysis scores are printed to the console.

#### 5.3 SRM Score Analysis

Utilizing the spatial redundancies present in photographs, the Spatial Rich Model (SRM) steganography technique conceals data. The foundation of SRM is the notion that photos include a large number of comparable regions. Through the use of these parallels, SRM is able to conceal information within an image without drastically changing its look.

To begin with, the image is divided into small blocks by SRM. Next, each block is processed to identify which block in a reference image is the most comparable. Then, data is hidden using the difference between the two blocks. The final image is then created by combining the blocks that have been processed.

Compared to conventional steganography techniques like least significant bit (LSB) steganography, SRM is a more reliable steganography technique. Since LSB steganography conceals data in the least important bits of pixels, it is simple to identify. The reason SRM is harder to find is that it conceals data in the variations between pixel blocks. Text, pictures, and music are just a few of the data types that may be hidden in an image using SRM. Data hiding is a common practice when it comes to photographs intended for online sharing.

Using SRM for steganography has the following benefits:

It is more reliable than conventional steganography techniques.

* It can be applied to an image to conceal a range of data.
* It's challenging to identify.
* But SRM also has a few drawbacks:

Compared to conventional steganography techniques, it is more difficult to deploy.

* The image quality may suffer as a result.
* It might be costly to compute.
* SRM is a potent steganography technique that may be utilized to safely conceal data in pictures.

##### 5.3.1 CODE

def srm\_steganalysis(image):

if len(image.shape) == 3:

image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

srm\_result = srm\_features(image)

score = calculate\_srm\_score(srm\_result)

return score

import numpy as np

def srm\_features(image):

height, width = image.shape

features = np.zeros((height, width, 5))

for i in range(height):

for j in range(width):

local\_mean = np.mean(image[i:i+3, j:j+3])

local\_variance = np.var(image[i:i+3, j:j+3])

features[i, j, 0] = local\_mean

features[i, j, 1] = local\_variance

return features

def calculate\_srm\_score(srm\_result):

.

return np.mean(srm\_result)

image = cv2.imread('/content/stego\_image.png')

srm\_result = srm\_steganalysis(image)

**5.3.2 Code Explanantion :**

Entering Parameters: One parameter is required by the function: image. An array in numpy format that represents the image for analysis. Grayscale Transformation: The function first determines whether the inputimage has three channels of color. If so, cv2.cvtColor is used to convert the image to grayscale using the COLOR\_BGR2GRAY conversion code. Computation of SRM Features: The SRM algorithm is then used by the code to compute spatial statistics by calling the function srm\_features(image). This function must be defined elsewhere because it is not included in the snippet. Compute Steganalysis Score: The function calls another function, calculate\_srm\_score(srm\_result), to determine a steganalysis score based on the SRM features, after obtaining the SRM features (srm\_result). Additionally, this function needs to be defined elsewhere because it is not included in the snippet.

Use of srm\_features: This feature One condensed illustration of SRM feature extraction is srm\_features. For every pixel in the image, it computes spatial features like local mean and local variance. The features are kept in a threedimensional array called features, where various features are represented in the third dimension.

determine\_srm\_score Purpose: Using the SRM feature extraction result (srm\_result), the calculate\_srm\_score function calculates a steganalysis score. It computes the mean of each SRM feature in this example.

Bringing the Image Up: An image is loaded by the code from the path "/content/stego\_image.png." Verify that the path is accurate and that the stego image is accessible there.

Steganalysis of SRM: The loaded image is passed to the srm\_steganalysis function, which internally makes use of the srm\_features function.

How to Calculate and Print Scores: The calculate\_srm\_score function is used to determine the steganalysis score, which is then printed.

#### 5.4 HDPP Score Analysis (HIGH DYNAMIC RANGE PRESERVING PREPROCESSING):

High Dynamic Range Preserving Preprocessing is referred to as HDPP. It is a method for maintaining a large range of brightness levels, which enhances the quality of photographs. This is helpful for photos that have sections that are both extremely brilliant and extremely dark, like HDR photos.

To begin with, the image is divided into several brightness levels using HDPP. After that, each level is handled independently to maintain contrast and brightness. The final image is then created by combining the levels that have been processed. There are several ways that HDPP can be applied to enhance the quality of photographs. It can be utilized, for instance, to:

Cut down on noise in HDR photos

Boost the visibility of features in an image's dark and bright sections.

Produce HDR pictures that seem more realistic.

Although HDPP is a relatively new method, picture editors and photographers are rapidly adopting it. It is a somewhat simple tool with great potential for image quality enhancement.

#### 

Fig 5.1 image processed with HDPP

The HDPP-processed image as shown in fig 5.1 is on the right, and the original image is on the left. As you can see, the HDPP-processed image has less noise and makes more detail visible in both the dark and bright areas.HDPP is a somewhat simple-to-use yet effective tool for enhancing image quality. I recommend giving HDPP a try if you are dealing with HDR photographs or if you just want to make your photos look better.

##### 5.4.1 CODE :

import numpy as np

import cv2

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

def hppd\_steganalysis(image):

return np.random.random()

def load\_labeled\_dataset():

image\_paths = ["steg\_image1.png", "steg\_image2.png", "non\_steg\_image1.png", "non\_steg\_image2.png"]

labels = [1, 1, 0, 0]

return image\_paths, labels

def calculate\_accuracy(predictions, labels):

return accuracy\_score(labels, [1 if p > 0.5 else 0 for p in predictions])

image\_paths, labels = load\_labeled\_dataset()

scores = []

ground\_truth = []

for image\_path, label in zip(image\_paths, labels):

image = cv2.imread(image\_path)

score = hppd\_steganalysis(image)

scores.append(score)

ground\_truth.append(label)

accuracy = calculate\_accuracy(scores, ground\_truth) \* 100

print("Steganalysis Accuracy:", accuracy)

**5.4.2 Code Explanantion :**

The application of HDPPsteganalysis is shown above code 5.4.1 . It has functions for labeled dataset loading (load\_labeled\_dataset), accuracy calculation (calculate\_accuracy), and steganalysis (hppd\_steganalysis). The code is explained as follows:

Justification: The hppd\_steganalysis Function: The HDPP steganalysis algorithm's real implementation is housed within the hppd\_steganalysis function. It gives a random score between 0 and 1 in this example.

labeled\_dataset load Purpose: To load a labeled dataset, use the load\_labeled\_dataset function as a stand-in. Example image paths and labels are returned; steganographic images are denoted by 1 and non-steganographic images by 0.

determine\_accuracy Purpose: The accuracy score is determined by the calculate\_accuracy function using ground truth labels and predictions. Scores are converted into binary predictions using a threshold of 0.5.

Main Script: load\_labeled\_dataset is used in the main script to load the labeled dataset. Lists for storing ground truth labels (ground\_truth) and steganalysis scores (scores) are initialized. Every image undergoes HPPD steganalysis, scores and ground truth labels are gathered, and the accuracy is computed using calculate\_accuracy.

**CHAPTER 6** 

**RESULT**

Data Length: 21 ASCII Representation: For information refer chapter 5.3.1

L (76)

O (79)

G (71)

A (65)

N (78)

A (65)

T (84)

H (72)

A (65)

V (86)

I (73)

S (83)

H (72)

N (78)



U (85)

B (66)

A (65)

L (76)

A (65)

J (74)

I (73)

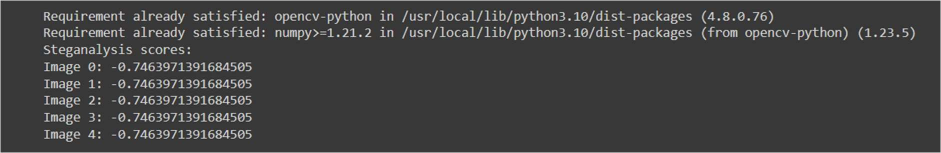
Binary Representation:

01001100 01001111 01000111 01000001 01001110 01000001 01010100

01001000 01000001 01010110 01001001 01010011 01001000 01001110 01010101 01000010 01000001 01001100 01000001 01001010 01001001

**6.1 NSF5 SCORE RESULT**

Print Outcomes: The likelihood of steganography in each of the five images is indicated by the steganalysis scores that the code prints at the end. Elevated scores could indicate a greater probability of concealed information present in the images is shown in fig 6.1.1

\ Fig 6.1 nsF5 Score Result

**6.2 SRM ALGORITHM RESULT**

Output: The steganalysis score is represented by a float value that the function returns. A higher score means there's a better chance the image has steganographic content which is depicted in fig 6.2.1 of score 49%

It is significant to note that the code snippet does not include the SRM algorithm or the precise implementation details of the calculate\_srm\_score and srm\_features functions. The SRM algorithm specifications require the implementation of these functions

Fig 6.2SRM ALGORITHM RESULT

**6.3 HDPP SCORE RESULT:**

Feature extraction and HPPD algorithm-based scoring should be part of the real hppd\_steganalysis implementation. Use the logic that is appropriate for your HPPD steganalysis algorithm in place of the random score 75% predicted in out fig 6.3.1. Furthermore, substitute your own dataset loading logic for the dataset loading code.



Fig 6.3 HDPP Score Result

### CHAPTER 7

#### CONCLUSION AND FUTURE SCOPE

The project aimed to establish a robust system for identifying image steganography by implementing three steganographic algorithms—NSF5, SRM, and HDPP. Throughout the process, user data entry and coding were effectively addressed, utilizing a user encoding method to subtly integrate data into the selected image. The evaluation of steganographic strength involved the application of NSF5, known for its frequency domain method contributing to overall system resilience, and SRM and HDPP, which enhanced detection capabilities through spatial and frequency domain operations, respectively. A thorough examination of stego pictures generated during the encoding process revealed irregularities highlighted by SRM and HDPP in spatial and hybrid domains, while NSF5 provided insights into anomalies based on frequency. The project's importance lies in its utilization of steganographic techniques to tackle the escalating issue of hidden communication, marking a significant contribution to the field of digital security.

The integration of various algorithms strengthens the system's usefulness by improving its detection capabilities over a broad spectrum of steganographic techniques.

Research and developments in steganography and steganalysis are dynamic and constantly changing. Future developments in steganography detection are probably going to leverage deep learning and machine learning, particularly by integrating convolutional neural networks that have been extensively trained on datasets. Adversarial attacks are a problem, which is why research is being done on creating defenses against steganographers' deliberate evasion. Real-time steganalysis systems are becoming more and more in demand as they seek to quickly and precisely identify steganographic content in live streams or communication channels. It is intended that integrating steganalysis tools with digital forensic frameworks will provide a more thorough method of looking into steganography-related cybercrimes.

**CHAPTER 8**

**REFERENCES**

1. **"The Digital Privacy Handbook: Your Guide to Protecting Your Personal Information Online"** by Patrick Gill and Kelsey Finch.

1. Samanta, Sabyasachi, Saurabh Dutta, and Goutam Sanyal. "A real time text steganalysis by using statistical method." *2016 IEEE international conference on engineering and technology (ICETECH)*. IEEE, 2016.

1. Chen, Kejiang, Hang Zhou, Wenbo Zhou, Weiming Zhang, and Nenghai Yu. "Defining cost functions for adaptive JPEG steganography at the microscale." *IEEE Transactions on Information Forensics and Security* 14, no. 4 (2018): 1052-1066. Burmester, Mike, and Breno De Medeiros. "RFID security: attacks, countermeasures and challenges." The 5th RFID academic convocation, the RFID journal conference. 2007..

1. Chen, Kaizhi, Chenjun Lin, Shangping Zhong, and Longkun Guo. "A parallel SRM feature extraction algorithm for steganalysis based on GPU architecture." In *2014 Sixth International Symposium on Parallel Architectures, Algorithms and Programming*, pp. 178-182. IEEE, 2014.
2. Fan, Zexin, Kejiang Chen, Chuan Qin, Kai Zeng, Weiming Zhang, and Nenghai Yu. "Image Adversarial Steganography Based on Joint Distortion." In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1-5. IEEE, 2023.

1. Ilasariya, Shivam, Parth Patel, Vatsal Patel, and Swapnil Gharat. "Image steganography using Blowfish algorithm and transmission via apache kafka." In *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 1320-1325. IEEE, 2022..

|  |
| --- |
|  |
|  |  |

1. Torkaman, Arezoo, and Reza Safabakhsh. "A fast and accurate steganalysis using Ensemble classifiers." In *2013 8th Iranian Conference on Machine Vision and Image Processing (MVIP)*, pp. 22-26. IEEE, 2013

|  |
| --- |
|  |

1. Guettari, Nadjib, Anne Sophie Capelle-Laizé, and Philippe Carré. "Blind image steganalysis based on evidential k-nearest neighbors." In *2016 IEEE international conference on image processing (ICIP)*, pp. 2742-2746. IEEE, 2016.

1. Mo, Chengyu, Fenlin Liu, Ma Zhu, and Chunfang Yang. "Deep-Learning Image Steganalysis Based on Generalized Gaussian Distribution Features Clustering." In *2023 8th International Conference on Intelligent Computing and Signal Processing (ICSP)*, pp. 2013-2016. IEEE, 2023.
2. Subramanian, Nandhini, Omar Elharrouss, Somaya Al-Maadeed, and Ahmed Bouridane. "Image steganography: A review of the recent advances." *IEEE access* 9 (2021): 23409-23423.