VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY
Department of Computer Engineering

V.E.S.
Since 1962

Project Report on

Data Security Using Multimedia Steganography

Submitted in partial fulfillment of the requirements of the degree

BACHELOR OF ENGINEERING IN **COMPUTER ENGINEERING**

By

Vishakha Singh, D12B-57

Manasi Sharma, D12B-54

Anushka Shirode, D12B-56

Project Mentor
Mr. Sanjay Mirchandani

University of Mumbai
(AY 2023-24)

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Department of Computer Engineering

V.E.S.
Since 1962

CERTIFICATE

This is to certify that the Mini Project entitled **"Data Security Using Multimedia Steganography "** is a bonafide work of **Vishakha Singh(D12B 57), Manasi Sharma(D12B 54), Anushka Shirode(D12B 56)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **"Bachelor of Engineering"** in **"Computer Engineering" .**

Sanjay Mirchandani

Digitally signed by Sanjay Mirchandani
DN: cn=Sanjay Mirchandani, o=V.E.S. Institute of Technology, ou=Assistant Professor, email=sanjay.mirchandani@ves.ac.in, c=IN
Date: 2024.04.09 19:25:08 +05'30'

(Mr. Sanjay Mirchandani)
Mentor

(Dr. Mrs. Nupur Giri)
Head of Department

(Dr. Mrs. J.M. Nair)
Principal

Mini Project Approval

This Mini Project entitled “**Data Security Using Multimedia Steganography”** by **Vishakha Singh(D12B 57), Manasi Sharma(D12B 54), Anushka Shirode(D12B 56)** is approved for the degree of **Bachelor of Engineering** in **Computer Engineering.**

Examiners

1……………………………………….
(Internal Examiner Name & Sign)

2……………………………………….
(External Examiner name & Sign)

Date: 01/04/2024

Place: Chembur, Mumbai

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

[illegible]

(Vishakha Singh - 57)

[illegible]

(Manasi Sharma - 54)

[illegible]

(Anushka Shirode - 56)

Date: 01/04/2024

ACKNOWLEDGEMENT

We are thankful to our college Vivekanand Education Society’s Institute of Technology for considering our project and extending help at all stages needed during our work of collecting information regarding the project.

It gives us immense pleasure to express our deep and sincere gratitude to Assistant Professor **Mr. Sanjay Mirchandani** (Project Mentor) for his kind help and valuable advice during the development of project synopsis and for his guidance and suggestions.

We are deeply indebted to Head of the Computer Department **Dr.(Mrs.) Nupur Giri** and our Principal **Dr. (Mrs.) J.M. Nair,** for giving us this valuable opportunity to do this project.

We express our hearty thanks to them for their assistance without which it would have been difficult in finishing this project synopsis and project review successfully.

We convey our deep sense of gratitude to all teaching and non-teaching staff for their constant encouragement, support and selfless help throughout the project work. It is a great pleasure to acknowledge the help and suggestion, which we received from the Department of Computer Engineering.

We wish to express our profound thanks to all those who helped us in gathering information about the project. Our families too have provided moral support and encouragement several times.

Index

List of Abbreviations

Abbreviation	**Full Form**
CNN	Convolutional Neural Networks
LSB	Least Significant Bit
GAN	Generative Adversarial Networks
SIFT	Scale-Invariant Feature Transform
SRM	Spatial Rich Model
MSE	Mean Squared Error
PSNR	Peak Signal-to-Noise Ratio
SSIM	Structural Similarity Index Measure
GMM	Gaussian Mixture Model
ReLU	Rectified Linear Unit
DCT	Discrete Cosine Transform

List of Figures

List of Tables

Abstract

Various pre-available tools like Steghide, OpenPuff, OpenStego, and Audacity have been explored for image and audio steganography. Steganalysis algorithms using the Python library, Shannon Entropy, Gaussian Mixture Model, Scale-Invariant Feature Transform, and Spatial Rich Model have been used. A comparison table comprising the various algorithms in combination with the pre-available tool is done to draw the necessary conclusions. The work aims to explore and compare the accuracy and loss values between various steganography techniques like Simple CNN, Batch Normalization, and ResNet101.

Chapter 1: Introduction

1.1 Introduction

In this study, data security techniques utilizing image steganography were explored, which involves concealing secret information within images. Steganography complements encryption by hiding data within innocuous-looking files, thereby enhancing confidentiality and security. Our research investigates various steganography tools and algorithms, aiming to understand their efficacy in concealing and detecting hidden information.

1.2 Motivation

The increasing need for secure communication and data transmission motivates our exploration of data security techniques. With the proliferation of digital communication channels and the growing threats of data breaches and cyberattacks, there's a pressing demand for robust methods to safeguard sensitive information. By leveraging steganography alongside encryption, the aim to enhance the security of digital media and confidential communication.

1.3 Problem Definition

The study addresses the pressing need for effective techniques to conceal and detect hidden data within images, leveraging the power of convolutional neural networks (CNNs), inbuilt libraries, and algorithms for steganalysis. This research seeks to explore the efficacy of CNNs, alongside inbuilt libraries and algorithms for steganalysis, in concealing and detecting hidden data within images, aiming to provide a comprehensive comparison of their performance and capabilities in addressing this critical security concern.

1.4 Existing Systems

Several existing steganography tools and algorithms are available for concealing and detecting hidden information within images. Tools like Steghide, OpenPuff, and OpenStego offer different approaches to hiding data within image files, while algorithms such as Shannon Entropy, Gaussian Mixture Model, SIFT, and SRM provide techniques for steganalysis - the detection of hidden data.

1.5 Lacuna of existing systems

Despite the availability of steganography tools and algorithms, there are several limitations and gaps in existing systems. Some tools may lack robust encryption or may be susceptible to detection by advanced steganalysis techniques. Additionally, existing algorithms may struggle to detect subtle changes introduced by sophisticated steganography methods, highlighting the need for more advanced detection techniques.

1.6 Relevance of the Project

Our project is highly relevant in the context of modern cybersecurity challenges. With the increasing sophistication of cyber threats, there's a growing need for innovative approaches to data security. By exploring data security techniques using image steganography, the aim is to contribute to the development of more secure communication protocols and data transmission methods. Our research has implications for various domains, including military and intelligence operations, digital forensics, and secure communication channels.

Chapter 2: Literature Survey

A. "Information Hiding Using Steganography" by Ritu Sindhu and Pragati Singh [1].

- The paper discusses multiple approaches to steganography.
- The merit of this paper is an in-depth discussion of various techniques. The paper approaches to get back the steganographic text by using steganographic key-based encryption and decryption.
- The demerits observed were the need to implement neural networks, layers, and filters to classify a standalone image- whether it has data hidden in it or not, as real-world data hiding does not follow the norm of key-based steganography.

B. "Image Steganography: A Review of the Recent Advances" by Nandhini Subramanian, Somaya Al-Maadeed, Ahmed Bouridane [2].

- Image steganography techniques encompass traditional methods like LSB substitution, Convolutional Neural Network (CNN)-based approaches, and General Adversarial Network (GAN)-based methods. Each category offers distinct advantages and challenges in concealing secret information within images.
- CNN-based methods offer notable advantages in steganography, as they utilize deep learning to enhance security and stego-image quality. By leveraging complex pattern recognition, these methods bolster resistance against detection, thereby improving the overall efficacy of hiding information. On the other hand, GAN-based techniques excel in generating realistic cover images, thereby enhancing the concealment of hidden data and potentially boosting the effectiveness of steganography.
- However, CNN-based approaches may suffer from the drawback of requiring substantial computational resources for encoding and decoding, limiting their applicability in real-time scenarios. Similarly, the training of GANs for steganography poses challenges due to instability and the necessity for meticulous configuration, which could impede their practical usability.

C. In "Digital Image Steganalysis: Current Methodologies and Future Challenges" by Wafa M. Eid, Sarah S. Alotaib, Hasna M. Alqahtani and Sahar Q. Saleh[3].

- The paper provides a comprehensive survey of state-of-the-art techniques in image steganography and steganalysis, covering spatial and transform domain methods, classical machine learning, and deep learning approaches, while also addressing challenges and future research directions.
- While CNNs exhibit superior performance over classical methods, they are yet to achieve robustness against steganographic algorithms. The paper emphasizes the need for further research, particularly in leveraging Generative Adversarial Networks(GANs) to develop steganalysis algorithms capable of detecting hidden messages in real-world images.

Chapter 3: Requirement Gathering for the Proposed System

3.1 Introduction to Requirement Gathering

Requirement gathering is a crucial phase in the development of any system, including those related to data security techniques using image steganography. This phase involves systematically collecting, analyzing, and documenting the needs and expectations of stakeholders to ensure that the final solution meets their requirements. In our project, requirement gathering involves understanding the desired functionalities, performance expectations, and constraints of the system from both technical and user perspectives.

3.2 Functional Requirements

Functional requirements specify the specific functionalities and features that the system must provide to meet the needs of stakeholders [4]:

- Ability to hide secret data within image files using steganography tools such as Steghide, OpenPuff, and OpenStego.
- Capability to detect hidden data within images using steganalysis algorithms like Shannon Entropy, Gaussian Mixture Model, SIFT, and SRM.
- Integration of neural network architectures such as Simple CNN, ResNet, and Batch Normalization for advanced steganalysis.
- Evaluation of steganographic image quality using metrics like MSE, PSNR, and SSIM.

3.3 Non-Functional Requirements

Non-functional requirements define the quality attributes and constraints that the system must adhere to[4]. In the project, non-functional requirements includes:

- Security: Ensuring that the system provides robust encryption and data concealment mechanisms to prevent unauthorized access to hidden information.
- Performance: The system should exhibit efficient performance in terms of speed and accuracy in hiding and detecting hidden data.
- Usability: Designing intuitive user interfaces and providing clear documentation to facilitate user interaction and understanding.
- Reliability: The system should be dependable and resilient, capable of handling varying workloads and operating conditions without compromising performance.
- Scalability: The system should be scalable to accommodate increasing data volumes and user demands over time.
- Compatibility: Ensuring compatibility with different operating systems, hardware configurations, and software environments.

3.4 Hardware, Software, Technology, and Tools Utilized

- Hardware: Standard computing hardware including processors, memory, and storage devices.
- Software: Programming languages such as Python for algorithm implementation, and libraries like Stegano for steganalysis.
- Technology: Utilization of machine learning frameworks for neural network implementation, image processing techniques for steganography, and statistical analysis methods for steganalysis.
- Tools: Steganography tools like Steghide, OpenPuff, and OpenStego, as well as analysis tools like Shannon Entropy, Gaussian Mixture Model, SIFT, and SRM.

3.5 Constraints

Constraints refer to limitations or restrictions that may impact the development or deployment of the system. In the project, constraints may include:

- Computational resources: Limited availability of computational resources such as processing power, memory, and storage, which may affect the performance of steganalysis algorithms and neural network training.
- Compatibility constraints: Ensuring compatibility with existing systems, file formats, and network protocols to facilitate integration and interoperability.
- Regulatory constraints: Compliance with legal and regulatory requirements related to data privacy, encryption standards, and intellectual property rights.
- Time and budget constraints: Completion of the project within predefined timeframes and budgetary constraints, which may influence the scope and complexity of the solution.

By addressing these requirements, constraints, and considerations, one can effectively plan and develop a robust system for data security using image steganography techniques.

Chapter 4: Proposed Design

4.1 Block diagram of the system

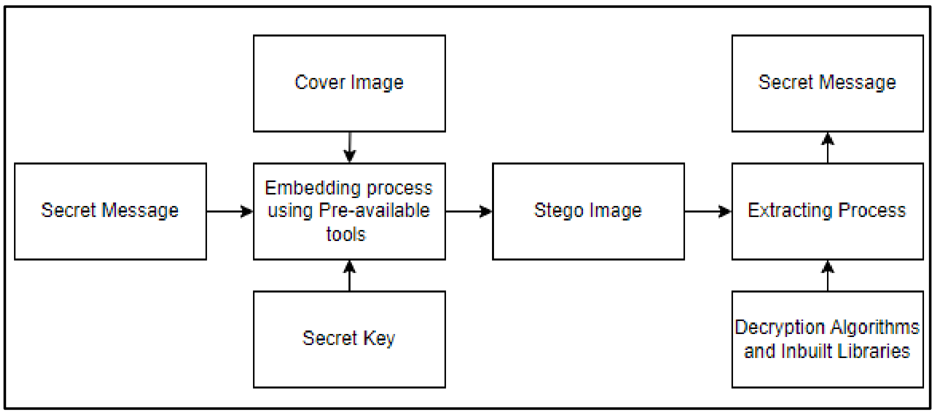

Cover Image
Secret Message
Secret Message
Embedding process using Pre-available tools
Stego Image
Extracting Process
Secret Key
Decryption Algorithms and Inbuilt Libraries

Fig 1. Block diagram

The block diagram consists of the following blocks:

1. Cover Image: The original image in which secret information is embedded. This could be any digital image, such as a photograph or a scanned document.
2. Secret Message: The confidential information that needs to be hidden within the cover image. This could be text, an image, or any other form of data.
3. Secret Key: A cryptographic key used to encrypt the secret message before embedding it into the cover image. This key ensures that only authorized parties can access and decrypt the hidden information.
4. Embedding Process: Using Available Tools or Models (e.g., CNN, Batch Normalization, ResNet101). This step involves embedding the secret message into the cover image using steganographic techniques. Various methods can be employed, such as Least Significant Bit (LSB) substitution or frequency domain techniques.
5. Stego Image: The resulting image after embedding the secret message. To the naked eye, the stego image appears identical to the original cover image, but it contains hidden information.
6. Secret Key (Again): The same secret key used during the embedding process is needed for decryption. It's crucial to keep this key secure and share it only with authorized parties.
7. Decryption Algorithm: This algorithm reverses the embedding process to extract the hidden information from the stego image using the secret key. The decryption algorithm should be designed to accurately recover the original secret message while minimizing the chances of false positives.
8. Extracting Process:This step involves applying the decryption algorithm to the stego image using the secret key.The algorithm extracts the hidden information and reconstructs the original secret message.The extracted message is then presented to the authorized recipient or processed further depending on the application.

4.2 Modular design of the system

The modular diagram consists of the following modules:

1. Data Preprocessing Module:
    - Handles dataset loading, preprocessing, and splitting into training and testing sets.
2. Feature Extraction Module:
    - Implements feature extraction algorithms like Scale-Invariant Feature Transform (SIFT), Gaussian Mixture Models (GMM), and Spatial Rich Model (SRM).
3. Deep Learning Model Module:
    - Includes implementations of deep learning models such as Simple CNN, ResNet, and models with Batch Normalization.
4. Evaluation and Reporting Module:
    - Generates reports and visualizations for analysis

4.3 Project Scheduling & Tracking using Timeline / Gantt Chart

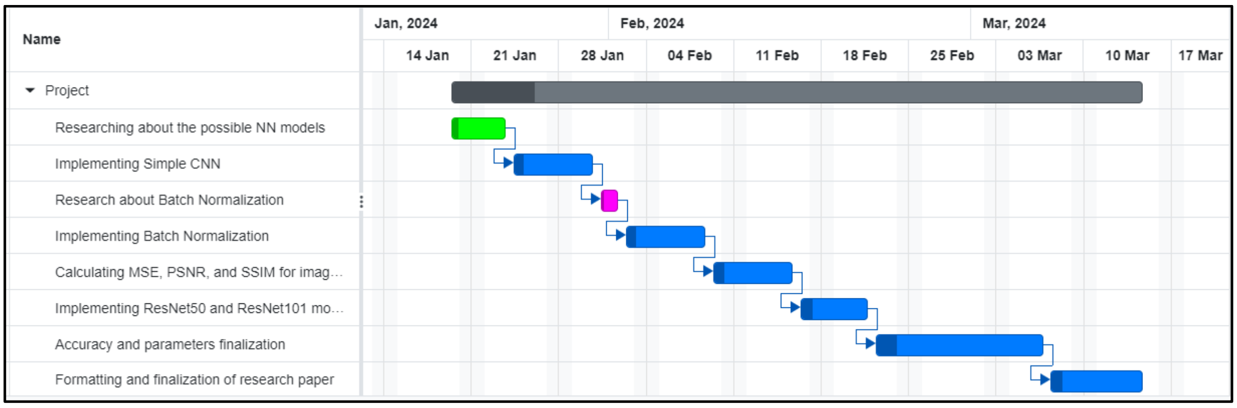


Fig 2. Gantt Chart

Chapter 5: Implementation of the Proposed System

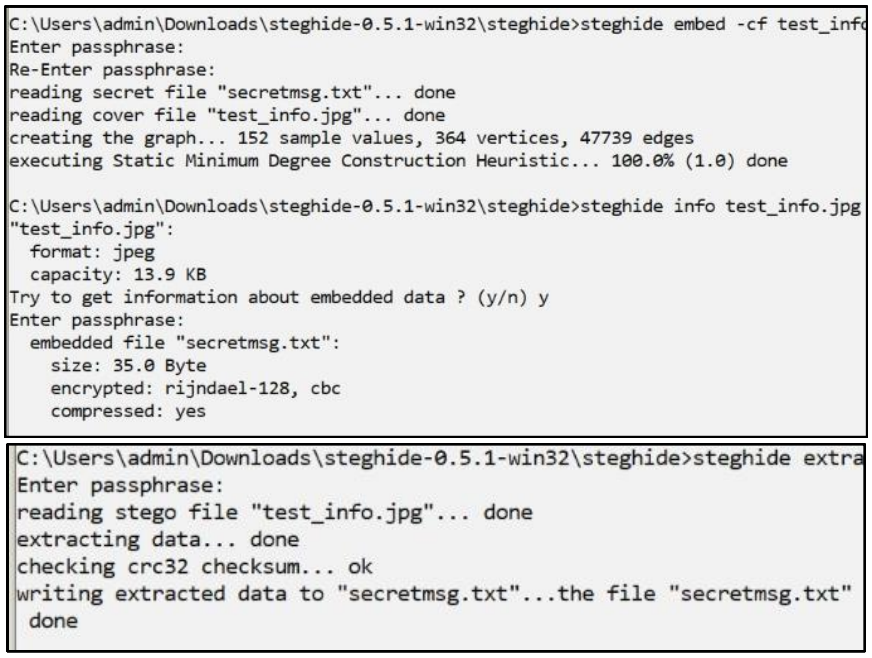
5.1 Methodology employed for development

The methodology employed includes:

1. Select or acquire a digital image (e.g., photograph, scanned document) to serve as the cover image.
2. Determine the confidential information that needs to be hidden within the cover image.
3. This could be text, an image, or any other form of data.
4. Generate a cryptographic key to encrypt the secret message.
5. Ensure the key is kept secure and shared only with authorized parties.
6. Utilize available tools or models (e.g., CNN, Batch Normalization, ResNet101) for embedding the secret message into the cover image.
7. Employ steganographic techniques such as Least Significant Bit (LSB) substitution or frequency domain techniques.
8. Obtain the resulting image after embedding the secret message.
9. Visually, the stego image should appear identical to the original cover image but contain hidden information.
10. Reuse the same secret key used during the embedding process for decryption.
11. Maintain the confidentiality and integrity of the secret key.
12. Design an algorithm specifically for decrypting the hidden information from the stego image using the secret key.
13. Ensure the decryption algorithm accurately recovers the original secret message while minimizing false positives.
14. Apply the decryption algorithm to the stego image using the secret key.
15. Extract the hidden information and reconstruct the original secret message.
16. Present the extracted message to authorized recipients or further process it according to the application's requirements.

A. Pre-Available Tools

1. Steghide
   Steghide is a free tool for hiding secret data inside image and audio files. It sneaks the data into the files without anyone noticing by tweaking the least important details. Figure 1 shows the process of encryption and decryption using Steghide [5].

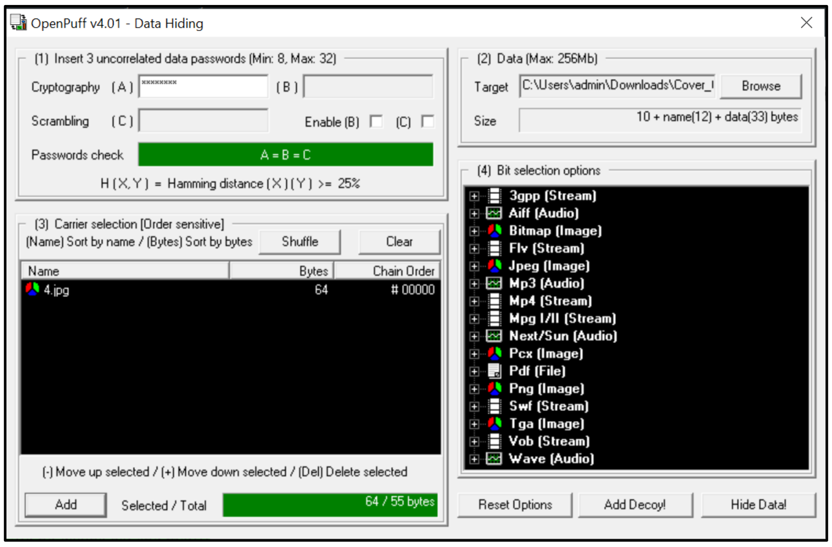


```
C:\Users\admin\Downloads\steghide-0.5.1-win32\steghide>steghide embed -cf test_infc
Enter passphrase:
Re-Enter passphrase:
reading secret file "secretmsg.txt"... done
reading cover file "test_info.jpg"... done
creating the graph... 152 sample values, 364 vertices, 47739 edges
executing Static Minimum Degree Construction Heuristic... 100.0% (1.0) done

C:\Users\admin\Downloads\steghide-0.5.1-win32\steghide>steghide info test_info.jpg
"test_info.jpg":
  format: jpeg
  capacity: 13.9 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "secretmsg.txt":
    size: 35.0 Byte
    encrypted: rijndael-128, cbc
    compressed: yes
```

```
C:\Users\admin\Downloads\steghide-0.5.1-win32\steghide>steghide extra
Enter passphrase:
reading stego file "test_info.jpg"... done
extracting data... done
checking crc32 checksum... ok
writing extracted data to "secretmsg.txt"...the file "secretmsg.txt"
 done
```

Fig 3. Steghide UI

2. OpenPuff
   OpenPuff is a free steganography tool for Windows that hides data in unique ways. Unlike most tools, it can split data across multiple carrier files (images, audio, video) creating a "carrier chain." OpenPuff offers strong security features like encryption and scrambling before hiding the data. It's known for its portability and multi-core processing for faster hiding and unhiding [6].



OpenPuff v4.01 - Data Hiding

(1) Insert 3 uncorrelated data passwords (Min: 8, Max: 32)
Cryptography (A) ******** (B)
Scrambling (C) Enable (B) ☐ (C) ☐
Passwords check: A = B = C
H (X,Y) = Hamming distance (X)(Y) >= 25%

(2) Data (Max: 256Mb)
Target: C:\Users\admin\Downloads\Cover_I
Size: 10 + name(12) + data(33) bytes

(3) Carrier selection [Order sensitive]
(Name) Sort by name / (Bytes) Sort by bytes

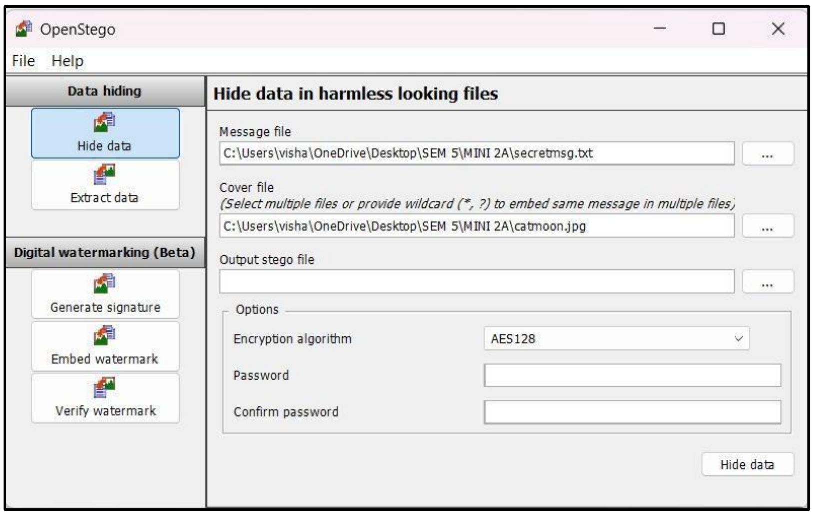
Name	Bytes	Chain Order
4.jpg	64	# 00000

(-) Move up selected / (+) Move down selected / (Del) Delete selected
Selected / Total: 64 / 55 bytes

(4) Bit selection options
3gpp (Stream)
Aiff (Audio)
Bitmap (Image)
Flv (Stream)
Jpeg (Image)
Mp3 (Audio)
Mp4 (Stream)
Mpg I/II (Stream)
Next/Sun (Audio)
Pcx (Image)
Pdf (File)
Png (Image)
Swf (Stream)
Tga (Image)
Vob (Stream)
Wave (Audio)

Fig 4. OpenPuff UI

3. OpenStego
   OpenStego is a free, open-source steganography tool written in Java and offers two main functions namely data hiding and watermarking. It uses plugins for steganographic algorithms, currently supporting Randomized LSB for hiding and Dugad's algorithm for watermarking [7].



Message file
C:\Users\visha\OneDrive\Desktop\SEM 5\MINI 2A\secretmsg.txt

Cover file
(Select multiple files or provide wildcard (, ?) to embed same message in multiple files)*
C:\Users\visha\OneDrive\Desktop\SEM 5\MINI 2A\catmoon.jpg

Output stego file

Options
Encryption algorithm: AES128
Password
Confirm password

Fig 5. OpenStego UI

B. Inbuilt Libraries and Mathematical Algorithms

1. Stegano Library (Python)
   The Stegano library spots hidden information in images by employing the Least Significant Bit (LSB) technique. This method involves altering the least important part of the pixel values to hide data within the image. It employs threshold-based detection, comparing the ratio of revealed data to the total image size with a predefined threshold (default value: 0.4) to determine steganography presence [8].

2. Shannon Entropy
   Shannon entropy, used in image steganalysis, detects hidden data by assessing the randomness of image regions. Natural images exhibit higher entropy, while steganography reduces randomness, especially in the Least Significant Bits (LSBs). Analysts identify areas with lower entropy, indicating potential hidden data, although some steganography methods aim to maintain randomness, making it a valuable but not foolproof tool [9].

3. Gaussian Mixture Model (GMM)
   GMM identifies statistical anomalies resulting from hidden data by modeling complex pixel value distributions in natural images. GMM detects deviations from these distributions, particularly in LSBs, where steganography disrupts natural patterns. GMM provides a potent tool for steganalysis by pinpointing statistical inconsistencies [10].

4. Scale-Invariant Feature Transform (SIFT)
   SIFT isn't typically used directly for steganalysis, but it can be a helpful pre-processing step. SIFT identifies key points in an image - areas with distinctive edges or corners. These key points are robust to scaling, rotation, and illumination changes, making them reliable features. Steganography can introduce subtle distortions that might affect SIFT key point locations or properties. By comparing SIFT key points in a

suspect image to those from a clean image, analysts might find inconsistencies. These inconsistencies could indicate tampered regions where data might be hidden [11].

5. Spatial Rich Model (SRM)
   SRM targets high-frequency details in images for steganalysis by analyzing statistics of neighboring noise residuals. It detects disruptions in natural patterns caused by steganography, particularly in LSBs, by examining changes in residual statistics. By leveraging pixel relationships, SRM identifies potential locations of hidden data, offering a potent tool for steganalysis by analyzing image statistic alterations [12].

5.2 Algorithms and flowcharts for the respective models developed

A. Stegano Library of Python

The algorithm is as follows :

1. Import Library: The code imports the lsb module from the stegano library, which provides functions for LSB steganography operations.
2. Function Definition: The detect_steganography function takes an image path (image_path) as input along with an optional parameter threshold (default is 0.4). This threshold determines the ratio of hidden data to total data, above which steganography is considered to be detected.
3. Steganography Detection:The function attempts to reveal the LSB data hidden in the cover image using lsb.reveal(image_path).It calculates the length of the revealed data (data_length) and the size of the image (image_size) using lsb.image_size(image_path).The ratio of hidden data to total data is computed as ratio = data_length / image_size.If the ratio exceeds or equals the threshold, steganography is considered to be detected, and the function returns True. Otherwise, it returns False.
4. Exception Handling: The function includes exception handling to catch any errors that may occur during the reveal operation. If the operation fails, indicating that there is no hidden data, the function returns False.
5. Usage: The path to the image file (image_path) is specified, and the detect_steganography function is called with this image path. Based on the return value, the code prints whether steganography is detected or not.

B. Shannon Entropy

The algorithm of Shannon Entropy is as follows :

1. Import Libraries: The code imports necessary libraries such as cv2 (OpenCV) and numpy (for numerical operations).
2. Function Definition: The detect_steganography function takes an image path (image_path) as input along with optional parameters:
   entropy_threshold: Threshold for Shannon entropy of pixel intensities (default is 7.0).
   spatial_threshold: Threshold for spatial entropy (default is 0.02).

3. Image Processing: The function reads the image using OpenCV's cv2.imread function and converts it to grayscale using cv2.cvtColor.
4. Entropy Calculation: The histogram of pixel intensities is computed using cv2.calcHist, and then normalized. Shannon entropy of the histogram is calculated using the formula:
   entropy = -np.sum(histogram * np.log2(histogram + 1e-10)**)**
   This formula calculates the entropy of the image based on the distribution of pixel intensities.
5. Spatial Analysis: Spatial analysis is performed by calculating the absolute differences between adjacent pixels in the grayscale image using np.diff. The mean of these differences is calculated (spatial_entropy).
6. Detection: If both the Shannon entropy and spatial entropy exceed their respective thresholds, steganography is considered to be detected.
7. Usage: The path to the image file (image_path) is specified. The detect_steganography function is called with the image path. Based on the return value, the code prints whether steganography is detected or not.

C. Gaussian Mixture Model (GMM)

The algorithm of GMM is as follows :

1. Import Libraries: The code imports necessary libraries such as cv2 (OpenCV), numpy (for numerical operations), and GaussianMixture from sklearn.mixture.
2. Function Definition: The detect_steganography function takes an image path (image_path) as input along with optional parameters num_components and threshold. It aims to detect whether steganography is present in the given image.
3. Image Processing: The function reads the image using OpenCV's cv2.imread function and converts it to grayscale using cv2.cvtColor.
   The grayscale image is then reshaped into a 1D array (reshaped_image) for further processing.
4. Gaussian Mixture Model (GMM): A Gaussian Mixture Model (GMM) with the specified number of components (num_components) is created using GaussianMixture from sklearn.mixture.
   The GMM is trained on the reshaped grayscale image using the Expectation-Maximization algorithm (gmm.fit(reshaped_image)).
5. Detection: The weights and means of the Gaussian components (weights and means) are obtained from the trained GMM. The absolute difference between the means of the Gaussian components (mean_diff) is calculated. If the mean difference exceeds the specified threshold (threshold), steganography is considered to be detected, and the function returns True. Otherwise, it returns False.
6. Usage: The path to the image file (image_path) is specified. The detect_steganography function is called with the image path and optional parameters. Based on the return value, the code prints whether steganography is detected or not.

D. Scale-Invariant Feature Transform (SIFT)

The algorithm of SIFT is as follows :

1. Image Loading and Preprocessing: The code loads two images: a training image (image1) and a testing image (image2).The training image is converted to grayscale (training_gray), while the testing image is kept in color for visualization purposes (test_image).
2. SIFT Keypoint Detection: SIFT keypoints and descriptors are computed for both the training and testing images using the cv2.xfeatures2d.SIFT_create() function. Keypoints represent distinctive image features, while descriptors encode information about the keypoints' local image neighborhoods.
3. Visualization of Keypoints: The code draws keypoints on the training image, both with and without their sizes, using cv2.drawKeypoints().These visualizations help understand the locations and scales of the keypoints detected in the image.
4. Keypoint Matching: A brute-force matcher (cv2.BFMatcher) is created to match keypoints between the training and testing images.Matching is performed based on the Euclidean distance between descriptors.Matches are sorted based on their distances.
5. Visualization of Matching Points: The code generates a visualization (result) showing the best matching points between the training and testing images.These matching points indicate areas of similarity between the images.
6. Steganography Detection: The algorithm checks if the number of keypoints detected in the training and testing images is equal.If the number of keypoints is equal, it concludes that steganography is not detected. Otherwise, it indicates the detection of steganography.
7. Output: The code displays the training and testing images, visualizations of keypoints, and the best matching points.It also prints the number of keypoints detected in each image and the number of matching keypoints between the training and testing images.

E. Spatial Rich Model (SRM)

The algorithm of SRM is as follows :

1. Import Libraries: The code imports necessary libraries such as cv2 (OpenCV), numpy (for numerical operations), and matplotlib.pyplot (for visualization).
2. Gabor Filter Function: The apply_gabor_filter function applies a Gabor filter to the input image with specified parameters such as frequency (frequency) and orientation (theta). It uses cv2.getGaborKernel to generate the Gabor kernel and applies it to the image using cv2.filter2D.
3. Visualization Function: The visualize_srm function visualizes the original image and its corresponding SRM images obtained by applying Gabor filters with different frequencies and orientations. It plots the original image and multiple SRM images in a grid layout using Matplotlib.
4. Main Function: The main function loads the original and steganographic images (assumed to be grayscale), sets the parameters for Gabor filters (frequencies and

orientations), and calls the visualize_srm function to visualize the SRM images for both original and steganographic images.

5. Usage: The code is executed within the if __name__ == '__main__': block, which ensures that the main function is called only when the script is run directly, not when it's imported as a module.

F. SimpleCNN

The algorithm of SimpleCNN as shown in Fig 6 is as follows:

1. Create Sequential Model: Initialize a sequential model object.
2. Add Convolutional Layers:
    - Add a 2D convolutional layer with 32 filters, a kernel size of (3, 3), ReLU activation function, and input shape (64, 64, 3).
    - Add a max-pooling layer with a pool size of (2, 2).
    - Add a dropout layer with a dropout rate of 0.25 to prevent overfitting.
    - Add another 2D convolutional layer with 128 filters, a kernel size of (3, 3), and ReLU activation function to increase model complexity.
    - Add another max-pooling layer with a pool size of (2, 2).
    - Add another dropout layer with a dropout rate of 0.25.
3. Flatten Layer: Flatten the output from the convolutional layers to prepare for the fully connected layers.
4. Add Fully Connected Layers:
    - Add a dense layer with 64 units, ReLU activation function, and L2 regularization with a regularization parameter of 0.01 to prevent overfitting.
    - Add a dropout layer with a dropout rate of 0.5.
5. Output Layer:
    - Add a dense layer with 1 unit and sigmoid activation function for binary classification.
6. Compile the Model:
    - Compile the model using the Adam optimizer with a learning rate of 0.001.
    - Use binary cross-entropy as the loss function since it's a binary classification problem.
    - Monitor accuracy as the evaluation metric.
7. Train the Model:
    - Fit the model to the training data for 100 epochs with a batch size of 64.
    - Use the validation data for validation during training.
8. Evaluate the Model:
    - Evaluate the trained model on both the training and test datasets.
    - Retrieve the accuracy for both datasets.
9. Print Results:
    - Print the training and test accuracies obtained from the evaluation.

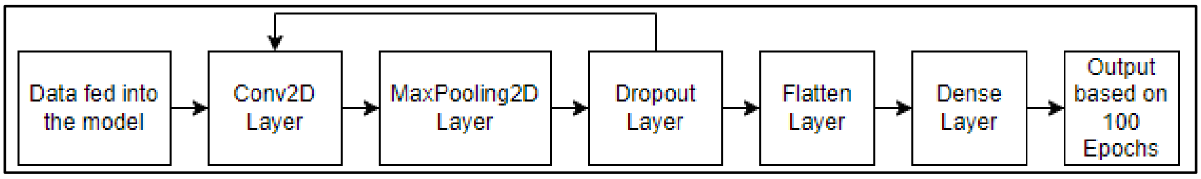


Fig 6. Flowchart for Simple CNN

G. ResNet101

The algorithm of ResNet101as shown in Fig 7 is as follows:

1. Load Pre-Trained ResNet101 Model:Load the pre-trained ResNet101 model without its fully connected layers. This pre-trained model serves as a feature extractor to capture high-level features from input images.
2. Freeze Pre-Trained Layers: Freeze the weights of the pre-trained ResNet101 layers to prevent them from being updated during training. This is done to retain the learned features.
3. Create New Model Architecture: Define a new neural network architecture on top of the pre-trained ResNet101 model. This includes adding convolutional layers, batch normalization layers, max-pooling layers, dropout layers, and dense layers.
4. Compile the Model: Compile the model with specified optimizer, loss function, and evaluation metrics. In this case, the Adam optimizer with a learning rate of 0.001 is used, and binary cross-entropy is chosen as the loss function. Accuracy is used as the evaluation metric.
5. Train the Model: Train the compiled model using training data (X_train, y_train_binary) for a specified number of epochs and batch size. Validation data (X_test, y_test_binary) is also provided for evaluating the model's performance during training.
6. Evaluate the Model: After training, evaluate the model's performance on both training and test datasets using the evaluate method. This calculates the loss value and accuracy of the model on the given data.
7. Print Results: Print the training and test accuracies obtained from the evaluation step.

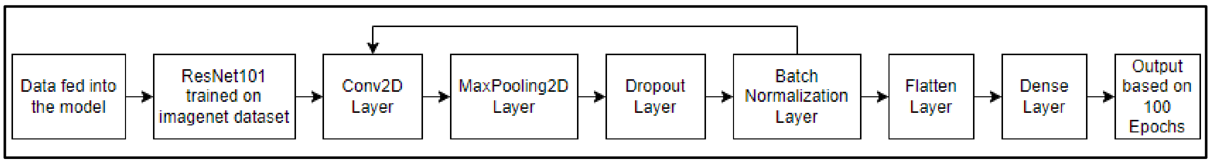


Fig 7. Flowchart for ResNet101

H. Batch Normalization

The algorithm of Batch Normalization as shown in Fig 8 is as follows :

1. Create Sequential Model: Initialize a sequential model.
2. Add Convolutional Layers with ReLU Activation: Add a Conv2D layer with 32 filters, kernel size (3, 3), and 'relu' activation function. Specify the input shape as (6, 64, 3) for the first layer.

3. Add Batch Normalization: Add a BatchNormalization layer after each convolutional layer. Batch normalization helps in stabilizing and accelerating the training process.
4. Add MaxPooling Layers: Add MaxPooling2D layers with a pool size of (2, 2) after each convolutional layer to down-sample the feature maps.
5. Add Dropout Layers: Add Dropout layers after each max-pooling layer with a dropout rate of 0.25. Dropout is a regularization technique used to prevent overfitting by randomly dropping units (along with their connections) from the neural network during training.
6. Increase Model Complexity: Add another Conv2D layer with 128 filters and 'relu' activation to increase the model's complexity.
7. Flatten Layer: Add a Flatten layer to flatten the 2D feature maps into a 1D vector to feed into the dense layers.
8. Add Dense Layers with ReLU Activation: Add a Dense layer with 64 units and 'relu' activation. Also, include kernel regularization with L2 regularization strength of 0.01 to prevent overfitting.
9. Add Dropout Layer: Add a Dropout layer with a dropout rate of 0.5 after the dense layer to further regularize the network.
10. Output Layer: Add a Dense layer with a single unit and 'sigmoid' activation function for binary classification.
11. Compile the Model: Compile the model using the Adam optimizer with a learning rate of 0.001, binary cross-entropy loss function, and accuracy as the evaluation metric.
12. Train the Model: Train the model on training data (X_train, y_train_one_hot) for 30 epochs with a batch size of 64. Use validation data (X_test, y_test_one_hot) for validation during training.
13. Evaluate the Model: Evaluate the trained model on both training and test datasets to obtain training and test accuracies.
14. Print Results: Print the training and test accuracies obtained from the evaluation step.

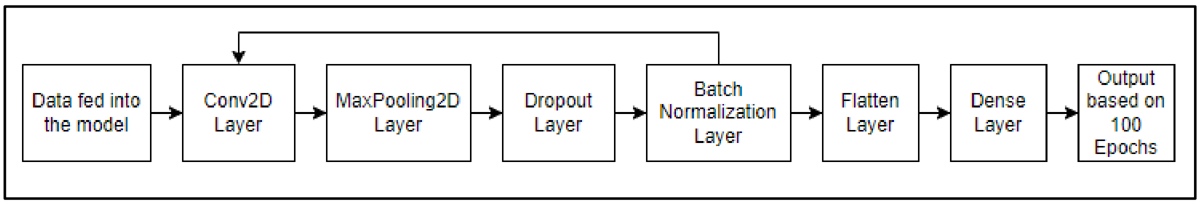


Fig 8. Flowchart for Batch Normalization

5.3 Datasets source and utilization

Dataset Source:
The dataset was created by generating 3000 images using the Discrete Cosine Transform (DCT) steganography technique. The dataset consists of 1500 cover images and 1500 stego images created by embedding information into the cover images using DCT steganography.

Utilization:

1. Training and Testing for Steganalysis Models: Researchers and developers can use this dataset to train and test steganalysis models. By providing both cover and stego images, the dataset enables the evaluation of the effectiveness of steganalysis algorithms in detecting hidden information in images manipulated using DCT steganography.
2. Benchmarking Steganalysis Techniques: The dataset can serve as a benchmark for evaluating the performance of different steganalysis techniques. Researchers can compare the accuracy, robustness, and efficiency of their steganalysis methods against existing approaches using this standardized dataset.
3. Algorithm Development: Developers can utilize the dataset to develop new steganalysis algorithms or improve existing ones. The diverse set of cover and stego images allows for comprehensive testing and validation of novel approaches aimed at detecting hidden information in images.
4. Educational Purposes: The dataset can be used in academic settings for educational purposes, such as teaching students about steganography, steganalysis, and image processing techniques. Students can gain hands-on experience with real-world data and explore different methodologies for analyzing and detecting hidden information in images.
5. Research on Steganography Security: The dataset facilitates research on steganography security by providing a large collection of images manipulated using the DCT steganography technique. Researchers can investigate the vulnerabilities of DCT steganography and develop countermeasures to enhance the security of digital content against unauthorized manipulation and information embedding.

Chapter 6: Testing of the Proposed System

6.1 Introduction to Testing

Testing is a crucial phase in the development lifecycle of any system, ensuring that it meets the specified requirements and functions as intended. In this chapter, testing methodologies are employed to evaluate the effectiveness and reliability of the proposed system for data security using image steganography.

6.2 Types of Tests Considered

Various types of tests have been considered to comprehensively evaluate the proposed system:

- Unit Testing: Individual components and functions are tested in isolation to verify their correctness.
- Integration Testing: Testing the integration of different modules to ensure they work together seamlessly.
- System Testing: Evaluating the system as a whole to verify that it meets the functional and non-functional requirements.
- Acceptance Testing: Conducted by end-users to validate that the system meets their expectations and requirements.
- Performance Testing: Assessing the system's performance under different conditions, including load testing and stress testing.
- Security Testing: Checking for vulnerabilities and ensuring that the system is resistant to attacks and unauthorized access.

6.3 Various Test Case Scenarios Considered

Test cases have been designed to cover a wide range of scenarios to ensure thorough testing of the proposed system:

1. Testing different steganography tools (e.g., Steghide, OpenPuff, OpenStego) with various input files and data types.
2. Evaluating the effectiveness of steganalysis algorithms (e.g., Shannon Entropy, Gaussian Mixture Model, SIFT) in detecting hidden information.
3. Testing the accuracy and performance of neural network models (e.g., Simple CNN, ResNet, Batch Normalization) in steganalysis.
4. Assessing the quality of steganographic images using metrics such as MSE, PSNR, and SSIM.
5. Conducting end-to-end testing to ensure that the system functions correctly from data hiding to detection.

6.4 Inference Drawn from the Test Cases

The test results provide valuable insights into the performance and reliability of the proposed system:

1. Identification of strengths and weaknesses in different steganography tools and algorithms.
2. Evaluation of the accuracy and efficiency of steganalysis techniques in detecting hidden information.
3. Assessment of the impact of varying parameters and configurations on the performance of neural network models.
4. Validation of the system's ability to conceal and detect hidden data while maintaining image quality and integrity.
5. Identification of areas for improvement and optimization to enhance the overall effectiveness of the system.

Analyzing the test results and drawing inferences enables informed decisions for refining and enhancing the proposed system to ensure optimal performance and security in real-world scenarios.

Chapter 7: Results and Discussion

7.1 Performance Evaluation measures

	MSE	**PSNR**	**SSIM**
Values	5.54	45.46	0.9981

Table 1. Evaluation Metrics

- The Mean Squared Error (MSE) is relatively low, indicating that the stego image has minimal distortion compared to the original image [13].
- The Peak Signal-to-Noise Ratio (PSNR) is relatively high (45.46 dB), indicating good quality of the stego image.
- The Structural Similarity Index (SSIM) is very close to 1 (0.9981), indicating a high degree of similarity between the original and stego images.

PSNR, SSIM, and MSE are used to evaluate the quality of steganography using quantitative measures. Less MSE and higher PSNR and SSIM values generally suggest that the hidden information is concealed more effectively with minimal visual distortion.

7.2 Input Parameters / Features considered

Image Dataset: A dataset comprising 3000 images was generated through a self-initiated process. Initially, 1500 images were collected, and subsequently subjected to steganography using Discrete Cosine Transform (DCT) technique, resulting in another set of 1500 images. Each image in the dataset was meticulously labeled to distinguish between its cover and stego forms.

Simple CNN

- Conv2D Layers, Batch Normalization, MaxPooling2D Layers, Dropout Layers, Flatten Layer, Dense Layers

Batch Normalization

- eps=1e-5, momentum=0.1, axis=-1, and center=True, scale=True [14].

ResNet Model

- Weights= imagenet
- Include_top = False
- Input_shape = (64, 64, 3)
- Loss function= binary_crossentropy
- optimizer=Adam optimizer
- learning rate= 0.001
- Conv2D, BatchNormalization, MaxPooling2D, Dropout, Flatten, and Dense layers

7.3 Graphical and statistical output

The table shows a comparison of the various algorithms used in combination with the pre-available tools. It indicates whether each algorithm can detect steganography when paired with a specific tool.

Out of the algorithms and libraries tested, Stegano Library of Python, Shannon Entropy, GMM, and SRM gave a negative result while SIFT gave a positive result.

	Steghide	**OpenStego**	**OpenPuff**
Stegano Library of Python	No	No	No
Shannon Entropy	No	No	No
GMM	No	No	No
SIFT	Yes	Yes	Yes
SRM	No	No	No

Table 2. Detection Comparison Table

In SIFT, feature detection entails recognizing unique key points within an image, which remain unchanged despite alterations in scale, rotation, or illumination. SIFT extracts key point descriptors by detecting local extrema in scale-space and determining their location, scale, and orientation based on gradient information. Figures 9 and 10 show feature and key point detection using SIFT [15].

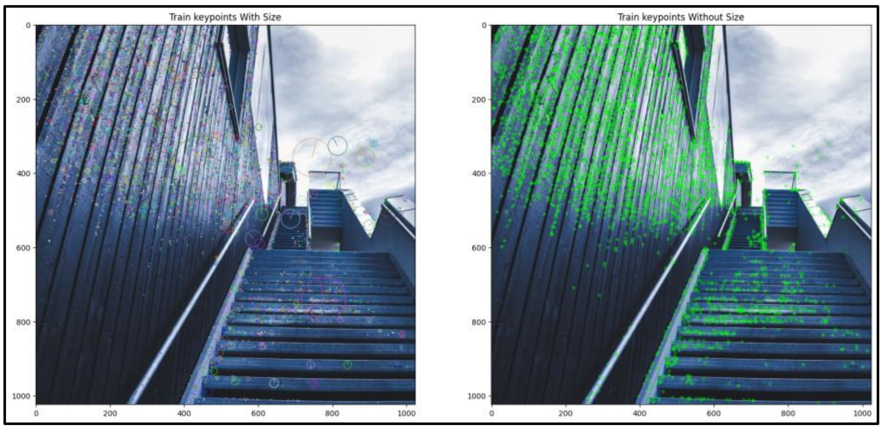


Fig 9. Feature Detection in SIFT

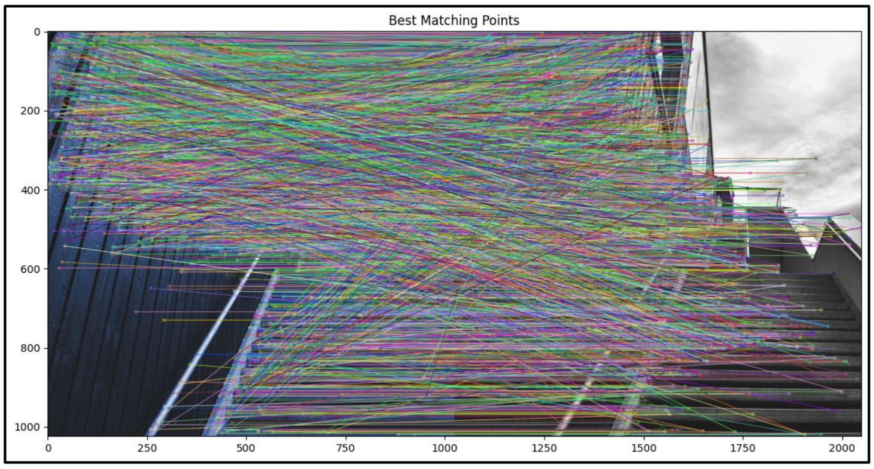


Fig 10. Key Points Detection in SIFT

Model Used	Split Ratio	Layers/ Parameters	Accuracy		Loss	
			Training	***Validation***	***Training***	***Validation***
Simple CNN	80:20	Conv2D Layers, Batch Normalization, MaxPooling2D Layers, Dropout Layers, Flatten Layer, Dense Layers	98.55%	98.58%	0.0867	0.1081
	75:25		98.26%	96.37%	0.1085	0.2089
	70:30		99.39%	98.10%	0.0705	0.1139
Batch Normalization	80:20	eps=1e-5, momentum=0.1, axis=-1, and center=True, scale=True.	99.32%	99.29%	0.3016	0.3099
	75:25		99.58%	98.53%	0.0602	0.1004
	70:30		99.23%	97.82%	0.0697	0.1020
ResNet Model	80:20	Weights= imagenet Include_top = False Input_shape = (64, 64, 3) Loss function= binary_crossentropy optimizer=Adam optimizer learning rate= 0.001 Conv2D, BatchNormalization, MaxPooling2D, Dropout, Flatten, and Dense layers.	97.44%	96.88%	0.1428	0.1582
	75:25		99.39%	98.75%	0.0586	0.0782
	70:30		97.97%	95.94%	0.2054	0.2770

Table 3. Comparison table for different CNN models

The models used for classification tasks include a Simple CNN, Batch Normalization, and a ResNet model, trained on various split ratios (80:20, 75:25, and 70:30) of training and validation data to generate accuracy and loss values.

The accuracy represents the percentage of correctly classified instances, while the loss measures the disparity between predicted and actual values during training, aiming to minimize this value to improve model performance. The variation in split ratios doesn't noticeably affect the accuracy and loss values.

The graphs in Figures 11 and 12 depict the accuracy of ResNet and Batch Normalization over 100 epochs on a 70:30 split dataset respectively. It consists of two lines: one tracking the accuracy on new, unseen data (validation accuracy), and the other depicting accuracy on the data the model was trained on (training accuracy).

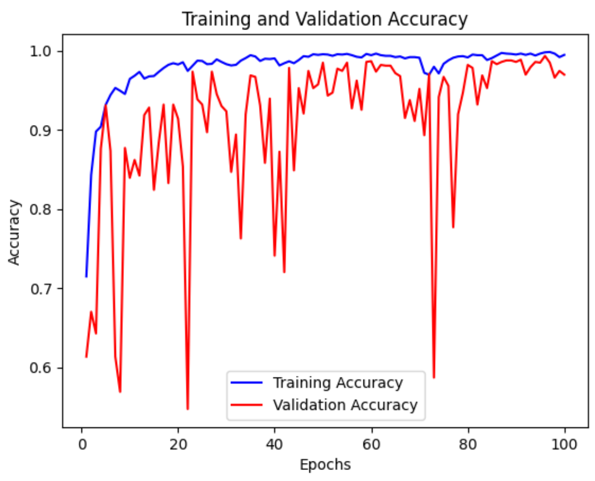


Fig 11. ResNet Accuracy

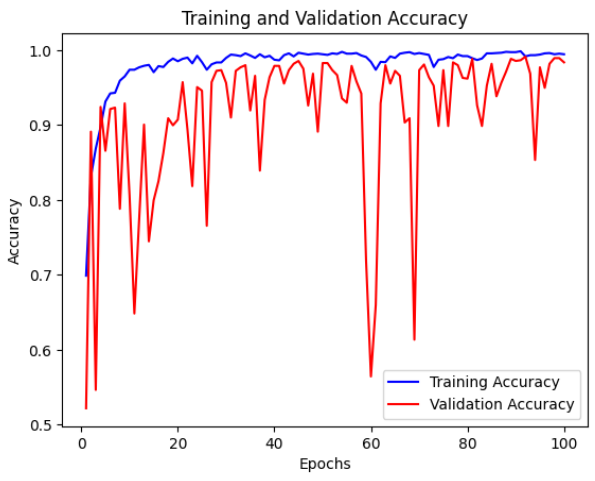


Fig 12. Batch Normalization Accuracy

7.4 Comparison of results with existing systems

In assessing ResNet101 for image steganography, its deep architecture and feature extraction capabilities hold promise, but its computational complexity warrants careful consideration. Batch normalization offers potential benefits in stabilizing training dynamics, yet its impact on steganographic outcomes requires thorough evaluation. CNNs, renowned for their versatility in image processing, present opportunities for effective feature extraction in steganography tasks, contingent upon careful architectural design and parameter tuning.

Balancing complexity and efficacy is crucial when integrating these models into steganographic systems, prioritizing both concealment and visual fidelity. Ultimately, a nuanced understanding of each model's strengths and limitations is essential for informed decision-making in deploying them for secure data concealment within images.

7.5 Inference drawn

Various steganography and steganalysis methods, including tools like Steghide, OpenStego, and OpenPuff, and algorithms such as GMM, SRM, Shannon Entropy, and SIFT, were used, alongside the integration of neural networks like Simple CNN, ResNet, and batch normalization for advanced steganalysis, highlighting the potential insights from further exploration of adversarial attacks and defenses for algorithm robustness.

Chapter 8: Conclusion

8.1 Limitations

Despite the comprehensive exploration and testing conducted in this study, several limitations were encountered:

1. Dependency on pre-available steganography tools and algorithms, which may have inherent limitations or vulnerabilities.
2. The performance of steganalysis techniques may vary depending on the complexity and sophistication of the steganography methods employed.
3. The proposed system's effectiveness may be influenced by external factors such as image quality, file format, and data size.
4. The accuracy of neural network models in steganalysis may be impacted by factors such as training data quality and model architecture.
5. Constraints such as computational resources and time limitations may restrict the scalability and efficiency of the system.

8.2 Conclusion

In conclusion, this study presents a comprehensive exploration of data security techniques using image steganography, encompassing the analysis of various steganography tools, steganalysis algorithms, and neural network models. Despite the challenges and limitations encountered, the study underscores the importance of integrating steganography with encryption for enhanced data security and confidentiality.

8.3 Future Scope

The future scope of this research includes:

1. Further refinement and optimization of steganography tools and algorithms to improve their robustness and effectiveness.
2. Exploration of advanced steganalysis techniques, including deep learning approaches and adversarial detection methods.
3. Investigation of novel applications and use cases for data security techniques using image steganography, including real-time communication and multimedia databases.
4. Collaboration with domain experts and stakeholders to address specific security challenges and requirements in various industries and sectors.
5. Continued research and development to stay abreast of emerging threats and technologies in the field of cybersecurity and data protection.

References

.

[1] Sindhu, R., & Singh, P. (2020). "Information Hiding using Steganography." International Journal of Engineering and Advanced Technology (IJEAT), 9(4), ISSN: 2249 – 8958 (Online).

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

[3] W. M. Eid, S. S. Alotaibi, H. M. Alqahtani and S. Q. Saleh, "Digital Image Steganalysis: Current Methodologies and Future Challenges," in IEEE Access, vol. 10, pp. 92321-92336, 2022, doi: 10.1109/ACCESS.2022.3202905.

[4] Functional Requirements, Non-Functional Requirements , alexsoft.com, URL (https://www. altexsoft.com/blog/)

[5] “Steghide Manual”, Steghide, URL (https://steghide.sourceforge.net/documentation/manpag e.php).

[6] “OpenPuff”, Wikipedia, URL (https://en.wikipedia.org/wiki/OpenPuff).

[7] “OpenStego”, OpenStego, URL (https://www.openstego.com/)

[8] “Stegano”, PyPi, URL (https://pypi.org/project/stegano/)

[9] Y. Wu, J. P. Noonan, and S. Agaian, "Shannon Entropy based Randomness Measurement and Test for Image Encryption," in IEEE Transactions on Information Forensics and Security, vol. 9, no. 3, pp. 459-470, March 2014.

[10] Scrucca, L. (2023). "Entropy-Based Anomaly Detection for Gaussian Mixture Modeling." Algorithms, 16(4), 195. https://doi.org/10.3390/a16040195

[11] “Introduction to SIFT(Scale Invariant Feature Transform)”, Medium, URL (https://mediu m.com/@deepanshut041/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40)

[12] Wang, Pengfei & Wei, Zhihui & Xiao, Liang. (2015). Pure spatial rich model features for digital image steganalysis. Multimedia Tools and Applications. 75. 10.1007/s11042-015-2521-9.

[13] Sara, U., Akter, M., & Uddin, M. S. (2019). "Image Quality Assessment through FSIM, SSIM, MSE, and PSNR—A Comparative Study." Journal of Computer and Communications, 7(3).

[14] Batch Normalisation, Introduction to convolutional neural networks, “d2l-ai”, URL(https://d2l.ai/chapter_convolutional-modern/batch-norm.html)

[15] SIFT Algorithm: How to Use SIFT for Image Matching in Python, “Analytics Vidhya”, URL(https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/)

[16] ResNet-101 convolutional neural network- resnet101, “mathworks” , URL (https://www.mat hworks .com/help/deeplearning/ref/resnet101.html)

[17] Steganography and Steganalaysis with CNN, “Medium”, URL (https://medium.com/geek culture/steganography-and-steganalaysis-with-cnn-9bd738dfed6f)

A Comprehensive Exploration Of Data Security Techniques Using Image Steganography

Vishakha Singh
Department Of Computer Engineering
V.E.S. Institute of Technology
Mumbai-40074, India
2021.vishakha.singh@ves.ac.in

Manasi Sharma
Department Of Computer Engineering
V.E.S. Institute of Technology
Mumbai-40074, India
2021.manasi.sharma@ves.ac.in

Anushka Shirode
Department Of Computer Engineering
V.E.S. Institute of Technology
Mumbai-40074, India
2021.anushka.shirode@ves.ac.in

Sanjay Mirchandani
Department Of Computer Engineering
V.E.S. Institute of Technology
Mumbai-40074, India
sanjay.mirchandani@ves.ac.in

***Abstract*—Various pre-available tools like Steghide, OpenPuff, OpenStego, and Audacity have been explored for image and audio steganography. Steganalysis algorithms using the Python library, Shannon Entropy, Gaussian Mixture Model, Scale-Invariant Feature Transform, and Spatial Rich Model have been used. A comparison table comprising the various algorithms in combination with the pre-available tool is done to draw the necessary conclusions. The work aims to explore and compare the accuracy and loss values between various steganography techniques like Simple CNN, Batch Normalization, and ResNet101.**

***Keywords*—Neural Networks, Steganalysis, Steganography.**

I. INTRODUCTION

Steganography involves concealing secret data within files like images or audio, while cryptography focuses on encrypting data to ensure its confidentiality, integrity, and authentication. Combining steganography with encryption improves security and finds applications in confidential communication and media databases, including military and intelligence operations[1]. Image steganography alters pixel values using encryption mechanisms of pre-available tools like Steghide, Openpuff, and Openstego. Mathematical algorithms and built-in libraries like Python, Shannon Entropy, Gaussian Mixture Model, SIFT, and SRM play crucial roles in steganography by analyzing data patterns and detecting hidden information within digital media, enhancing steganalysis and security.

In steganalysis, a hybrid CNN and ResNet architecture effectively differentiate between 'stego' and 'cover' images, optimizing parameters to enhance accuracy.

In steganography, measures like Mean Squared Error (MSE), Structural Similarity Index (SSIM), and Peak Signal to Noise Ratio (PSNR) are used to check how close the hidden image (stego) is to the original one (cover). These metrics help to understand how well the hidden information is preserved without making the image look noticeably different.

II. RELATED WORK

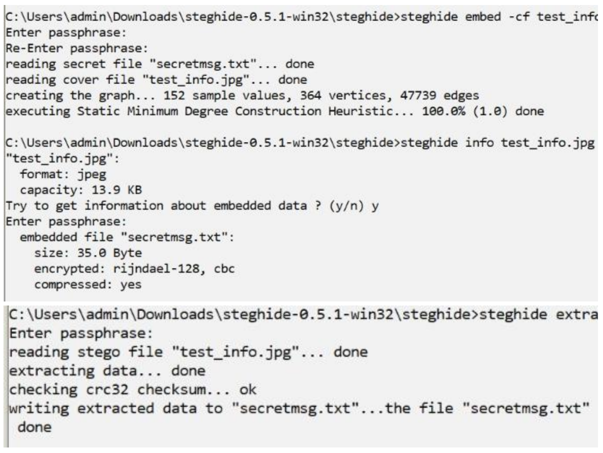
The paper [2] employs steganographic key-based encryption and decryption to retrieve steganographic text. It highlights the challenges of implementing neural networks, layers, and filters to classify standalone images for data-hiding detection, as real-world scenarios deviate from key-based steganography norms. Additionally, [3] discusses traditional LSB substitution, CNN-based, and GAN-based image steganography techniques. CNN-based methods leverage deep learning for improved security and stego image quality, while GAN-based techniques excel in generating realistic cover images, enhancing data concealment effectiveness. However, both approaches have drawbacks, such as CNN's high computational resource requirements and GAN's training challenges due to instability and configuration intricacies. [4] provides a survey of image steganography and steganalysis techniques, emphasizing the need for further research, especially while using GANs for developing steganalysis algorithms capable of detecting hidden messages in real-world images.

III. Methodology

A. Pre-Available Tools

1) Steghide

Steghide is a free tool for hiding secret data inside image and audio files. It sneaks the data into the files without anyone noticing by tweaking the least important details. [5]. Figure 1 shows the process of encryption and decryption using Steghide.



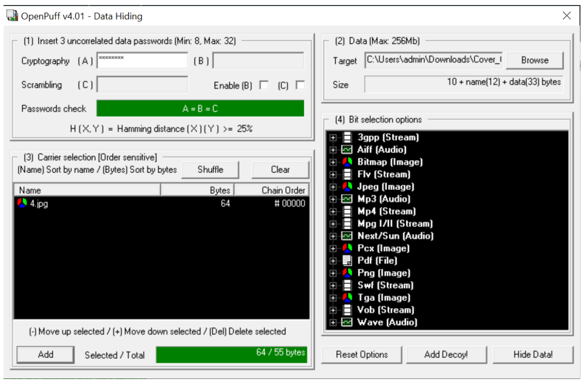
```
C:\Users\admin\Downloads\steghide-0.5.1-win32\steghide>steghide embed -cf test_inf
Enter passphrase:
Re-Enter passphrase:
reading secret file "secretmsg.txt"... done
reading cover file "test_info.jpg"... done
creating the graph... 152 sample values, 364 vertices, 47739 edges
executing Static Minimum Degree Construction Heuristic... 100.0% (1.0) done

C:\Users\admin\Downloads\steghide-0.5.1-win32\steghide>steghide info test_info.jpg
"test_info.jpg":
  format: jpeg
  capacity: 13.9 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "secretmsg.txt":
    size: 35.0 Byte
    encrypted: rijndael-128, cbc
    compressed: yes
```

```
C:\Users\admin\Downloads\steghide-0.5.1-win32\steghide>steghide extra
Enter passphrase:
reading stego file "test_info.jpg"... done
extracting data... done
checking crc32 checksum... ok
writing extracted data to "secretmsg.txt"...the file "secretmsg.txt"
 done
```

Fig. 1. Steghide

2) OpenPuff

OpenPuff is a free steganography tool for Windows that hides data in unique ways. Unlike most tools, it can split data across multiple carrier files (images, audio, video) creating a "carrier chain." OpenPuff offers strong security features like encryption and scrambling before hiding the data. It's known for its portability and multi-core processing for faster hiding and unhiding [6].



OpenPuff v4.01 - Data Hiding

(1) Insert 3 uncorrelated data passwords (Min: 8, Max: 32)
Cryptography (A) ******** (B)
Scrambling (C) Enable (B) ☐ (C) ☐
Passwords check: A = B = C
H (X,Y) = Hamming distance (X)(Y) >= 25%

(2) Data (Max: 256Mb)
Target: C:\Users\admin\Downloads\Cover_
Size: 10 + name(12) + data(33) bytes

(3) Carrier selection [Order sensitive]
(Name) Sort by name / (Bytes) Sort by bytes

Name	Bytes	Chain Order
4.jpg	64	# 00000

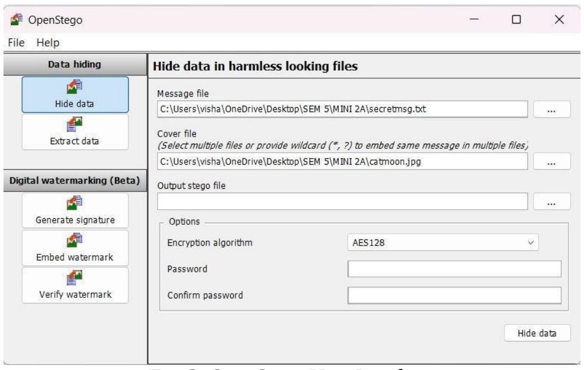
(-) Move up selected / (+) Move down selected / (Del) Delete selected
Selected / Total: 64 / 55 bytes

(4) Bit selection options
3gpp (Stream)
Aiff (Audio)
Bitmap (Image)
Flv (Stream)
Jpeg (Image)
Mp3 (Audio)
Mp4 (Stream)
Mpg I/II (Stream)
Next/Sun (Audio)
Pcx (Image)
Pdf (File)
Png (Image)
Swf (Stream)
Tga (Image)
Vob (Stream)
Wave (Audio)

Fig. 2. OpenPuff User Interface

3) OpenStego

OpenStego is a free, open-source steganography tool written in Java and offers two main functions namely data hiding and watermarking. It uses plugins for steganographic algorithms, currently supporting Randomized LSB for hiding and Dugad's algorithm for watermarking [7].



OpenStego

Hide data in harmless looking files

Message file: C:\Users\visha\OneDrive\Desktop\SEM 5\MINI 2A\secretmsg.txt

Cover file
(Select multiple files or provide wildcard (, ?) to embed same message in multiple files)*
C:\Users\visha\OneDrive\Desktop\SEM 5\MINI 2A\catmoon.jpg

Output stego file:

Options
Encryption algorithm: AES128
Password:
Confirm password:

Fig. 3. OpenStego User Interface

B. Mathematical Algorithms & Inbuilt Libraries

1) Stegano Library (Python)

The Stegano library spots hidden information in images by employing the Least Significant Bit (LSB) technique. This method involves altering the least important part of the pixel values to hide data within the image. It employs threshold-based detection, comparing the ratio of revealed data to the total image size with a predefined threshold (default value: 0.4) to determine steganography presence [8].

2) Shannon Entropy

Shannon entropy, used in image steganalysis, detects hidden data by assessing the randomness of image regions. Natural images exhibit higher entropy, while steganography reduces randomness, especially in the Least Significant Bits (LSBs). Analysts identify areas with lower entropy, indicating potential hidden data, although some steganography methods aim to maintain randomness, making it a valuable but not foolproof tool [9].

3) Gaussian Mixture Model (GMM)

GMM identifies statistical anomalies resulting from hidden data by modeling complex pixel value distributions in natural images. GMM detects deviations from these distributions, particularly in LSBs, where steganography disrupts natural patterns. GMM provides a potent tool for steganalysis by pinpointing statistical inconsistencies [10].

4) Scale-Invariant Feature Transform (SIFT)

SIFT isn't typically used directly for steganalysis, but it can be a helpful pre-processing step. SIFT identifies key points in an image - areas with distinctive edges or corners. These key points are robust to scaling, rotation, and illumination changes, making them reliable features. Steganography can introduce subtle distortions that might affect SIFT key point locations or properties. By comparing SIFT key points in a suspect image to those from a clean image, analysts might find inconsistencies. These inconsistencies could indicate tampered regions where data might be hidden [11].

5) Spatial Rich Model (SRM)

SRM targets high-frequency details in images for steganalysis by analyzing statistics of neighboring noise residuals. It detects disruptions in natural patterns caused by steganography, particularly in LSBs, by examining changes in residual statistics. By leveraging pixel relationships, SRM identifies potential locations of hidden data, offering a potent tool for steganalysis by analyzing image statistic alterations [12].

For testing these algorithms and libraries, images having text hidden in them via Steghide, OpenStego, and OpenPuff were used. These images were then given to the pre-available libraries and algorithms for detection.

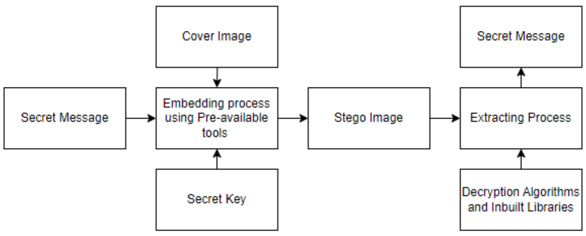

Cover Image
Secret Message
Embedding process using Pre-available tools
Stego Image
Extracting Process
Secret Message
Secret Key
Decryption Algorithms and Inbuilt Libraries

Fig 4. Flowchart for Steganography Detection

6) Audacity

While Audacity isn't designed for steganalysis, it can aid in the initial analysis and visualization of audio suspected to contain hidden data [13]. It offers waveform inspection for detecting unusual patterns, spectral analysis to identify hidden data in specific frequency ranges, and statistical analysis revealing changes in properties like RMS.

Audacity listens to sounds and shows them as pictures called spectrograms. Coagula hides secrets in audio, rendering them mostly blue when viewed, enabling the hiding of information perceptually.

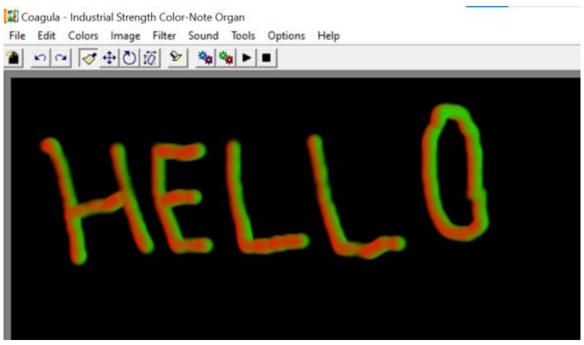

Coagula - Industrial Strength Color-Note Organ
File Edit Colors Image Filter Sound Tools Options Help
HELLO

Fig. 4. Coagula User Interface

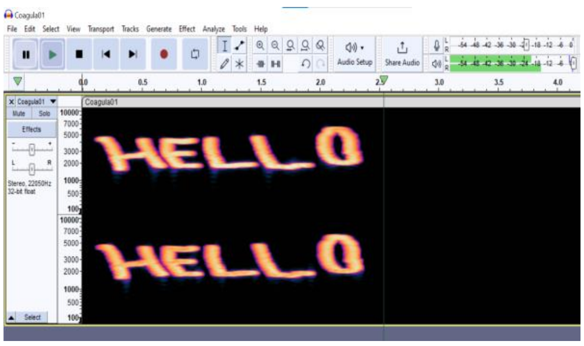

Coagula01
File Edit Select View Transport Tracks Generate Effect Analyze Tools Help
Audio Setup
Share Audio
HELLO
HELLO

Fig. 5. Audio file as spectrogram

IV. Implementation

PSNR, SSIM, and MSE are popular tools for assessing the quality of steganographic images created through the Discrete Cosine Transform (DCT) method.

The formula for calculating MSE is:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Here, n is the total number of pixels in the images while Y_i and $\hat{Y}_i$ are the pixel intensity of original and altered images respectively [14]. A lower MSE indicates less distortion and potentially better concealment of the hidden information.

PSNR measures how much clearer a signal is compared to the noise that might distort it, expressed in decibels. Higher PSNR values suggest higher image quality and less visible distortion. SSIM evaluates how similar the original and stego images are by considering their luminance, contrast, and structure. It offers a comprehensive assessment of perceptual quality. When SSIM values are higher, it suggests that the image characteristics are better preserved.

Neural Networks, namely CNN, are used for steganalysis.

1) Simple CNN
2) ResNet
3) BatchNormalisation

Simple CNN is a straightforward convolutional neural network (CNN) for image steganalysis, comprising Conv2D layers for feature extraction, Batch Normalization to standardize inputs and speed up training, MaxPooling2D layers to reduce spatial dimensions and extract dominant features, Dropout layers to prevent overfitting, a Flatten layer to convert the feature maps into a vector, and Dense layers for classification based on extracted features.

The ResNet101 is pre-trained on 101 layers using the image-net dataset. By removing the fully connected layers (include_top=False), the model essentially acts as a feature extractor, transforming input images into high-dimensional feature vectors while preserving important spatial information.

The proposed model uses a single-unit dense layer with sigmoid activation for binary classification. The model is trained using labeled data, where images are labeled as either containing hidden information (Cover) or not (Stego).

Batch Normalization ensures stable training dynamics and accelerates convergence by reducing internal covariate shifts by normalizing the activations of each layer. The parameters, specifically epsilon (eps), momentum, axis, center, and scale, define how Batch Normalization operates within the CNN model.

V. Results

Among tested steganography tools, Steghide offers strong encryption for hiding data, OpenStego provides a user-friendly interface, and OpenPuff stands out for its advanced steganographic techniques.

The table shows a comparison of the various algorithms used in combination with the pre-available tools. It indicates whether each algorithm can detect steganography when paired with a specific tool.

Out of the algorithms and libraries tested, Stegano Library of Python, Shannon Entropy, GMM, and SRM gave a negative result while SIFT gave a positive result.

TABLE I. Detection Comparison Table

	Steghide	**OpenStego**	**OpenPuff**
Stegano Library of Python	No	No	No
Shannon Entropy	No	No	No
GMM	No	No	No
SIFT	Yes	Yes	Yes
SRM	No	No	No

PSNR, SSIM, and MSE are used to evaluate the quality of steganography using quantitative measures. Less MSE and higher PSNR and SSIM values generally suggest that the hidden information is concealed more effectively with minimal visual distortion.

TABLE II. Evaluation metrics

	MSE	**PSNR**	**SSIM**
Values	5.54	45.46	0.9981

In SIFT, feature detection entails recognizing unique key points within an image, which remain unchanged despite alterations in scale, rotation, or illumination. SIFT extracts key point descriptors by detecting local extrema in scale-space and determining their location, scale, and orientation based on gradient information.

Figures 6 and 7 show feature and key point detection using SIFT.

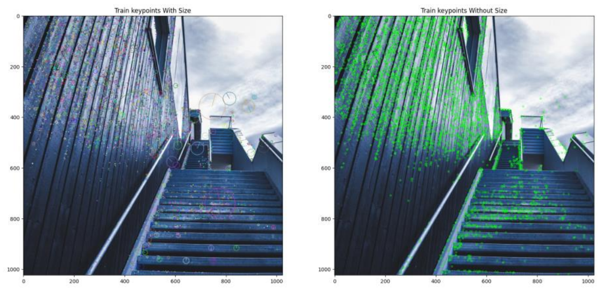


Fig 6. Feature Detection in SIFT

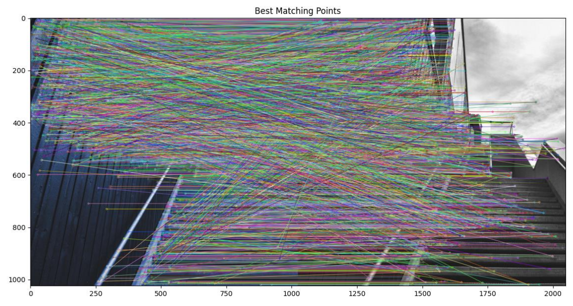


Fig 7. Key Points Detection in SIFT

TABLE III. Comparison table for different CNN models

Model Used	**Split Ratio**	**Layers/ Parameters**	**Accuracy**		**Loss**	
			Training	***Validation***	***Training***	***Validation***
Simple CNN	80:20	Conv2D Layers, Batch Normalization, MaxPooling2D Layers, Dropout Layers, Flatten Layer, Dense Layers	98.55%	98.58%	0.0867	0.1081
	75:25		98.26%	96.37%	0.1085	0.2089
	70:30		99.39%	98.10%	0.0705	0.1139
Batch Normalization	80:20	eps=1e-5, momentum=0.1, axis=-1, and center=True, scale=True.	99.32%	99.29%	0.3016	0.3099
	75:25		99.58%	98.53%	0.0602	0.1004
	70:30		99.23%	97.82%	0.0697	0.1020
ResNet Model	80:20	Weights= imagenet Include_top = False Input_shape = (64, 64, 3) Loss function= binary_crossentropy optimizer=Adam optimizer learning rate= 0.001 Conv2D, BatchNormalization, MaxPooling2D, Dropout, Flatten, and Dense layers.	97.44%	96.88%	0.1428	0.1582
	75:25		99.39%	98.75%	0.0586	0.0782
	70:30		97.97%	95.94%	0.2054	0.2770

The models used for classification tasks include a Simple CNN, Batch Normalization, and a ResNet model, trained on various split ratios (80:20, 75:25, and 70:30) of training and validation data to generate accuracy and loss values.

The accuracy represents the percentage of correctly classified instances, while the loss measures the disparity between predicted and actual values during training, aiming to minimize this value to improve model performance. The variation in split ratios doesn't noticeably affect the accuracy and loss values.

The graphs in Figures 8 and 9 depict the accuracy of ResNet and Batch Normalization over 100 epochs on a 70:30 split dataset respectively. It consists of two lines: one tracking the accuracy on new, unseen data (validation accuracy), and the other depicting accuracy on the data the model was trained on (training accuracy).

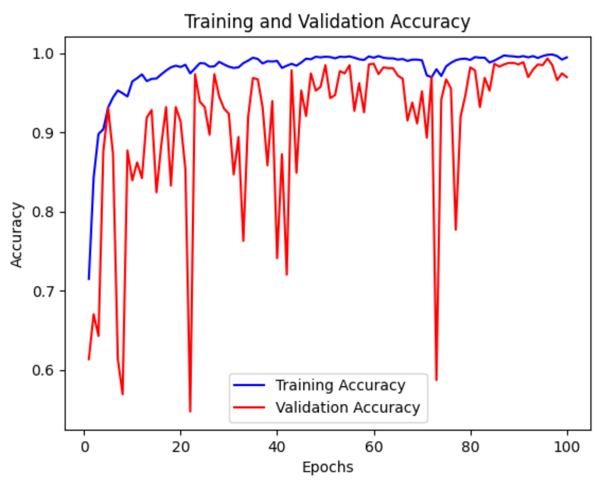


Fig 8. ResNet Accuracy

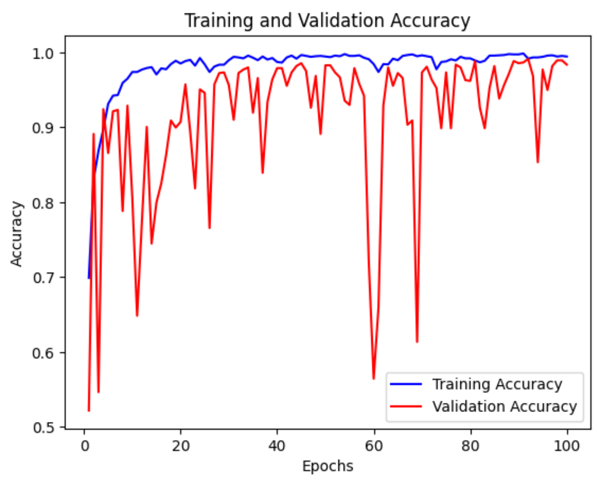


Fig 9. Batch Normalization Accuracy

From the comparison table, it is visible that Batch Normalization fits best to the requirements yielding an accuracy of 99.32% and 99.29% with a loss of 0.3016 and 0.3099 on training and validation sets, respectively, in the 80:20 split ratio.

VI. Conclusion And Future Scope

The paper discussed various steganography and steganalysis techniques, exploring tools such as Steghide, OpenStego, and OpenPuff, as well as various algorithms like GMM, SRM, Shannon Entropy, and SIFT. Additionally, a thorough discussion about the integration of neural network architectures like Simple CNN, ResNet, and batch normalization techniques for advanced steganalysis has been carried out.

Future research could focus on enhancing steganalysis techniques by leveraging deep learning approaches with more sophisticated neural network architectures. Moreover, exploring the potential of adversarial attacks and defenses within the context of steganography could provide valuable insights into the robustness of existing algorithms.

References

[1] I. Kich, E. Ameur, Y. Taouil, and A. Benhfid, "Image Steganography by Deep CNN Auto-Encoder Networks," IJATCSE.

[2] Sindhu, R., & Singh, P. (2020). "Information Hiding using Steganography." International Journal of Engineering and Advanced Technology (IJEAT), 9(4), ISSN: 2249 – 8958 (Online).

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

[4] W. M. Eid, S. S. Alotaibi, H. M. Alqahtani and S. Q. Saleh, "Digital Image Steganalysis: Current Methodologies and Future Challenges," in IEEE Access, vol. 10, pp. 92321-92336, 2022, doi: 10.1109/ACCESS.2022.3202905.

[5] "Steghide Manual", Steghide, URL (https://steghide.sourceforge.net/documentation/manpage.php).

[6] "OpenPuff", Wikipedia, URL (https://en.wikipedia.org/wiki/OpenPuff).

[7] "OpenStego", OpenStego, URL (https://www.openstego.com/)

[8] "Stegano", PyPi, URL (https://pypi.org/project/stegano/)

[9] Y. Wu, J. P. Noonan, and S. Agaian, "Shannon Entropy based Randomness Measurement and Test for Image Encryption," in IEEE Transactions on Information Forensics and Security, vol. 9, no. 3, pp. 459-470, March 2014.

[10] Scrucca, L. (2023). "Entropy-Based Anomaly Detection for Gaussian Mixture Modeling." Algorithms, 16(4), 195. https://doi.org/10.3390/a16040195

[11] "Introduction to SIFT(Scale Invariant Feature Transform)", Medium, URL (https://medium.com/@deepanshut041/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40)

[12] Wang, Pengfei & Wei, Zhihui & Xiao, Liang. (2015). Pure spatial rich model features for digital image steganalysis. Multimedia Tools and Applications. 75. 10.1007/s11042-015-2521-9.

[13] "Audacity", Forum Audacity Team, URL (https://forum.audacityteam.org/t/steganography/50214)

[14] Machine learning: an introduction to mean squared error and regression lines, URL (https://www.freecodecamp.org/news/machine-learning-mean-squared-error-regression-line-c7dde9a26b93/)

[15] Sara, U., Akter, M., & Uddin, M. S. (2019). "Image Quality Assessment through FSIM, SSIM, MSE, and PSNR—A Comparative Study." Journal of Computer and Communications, 7(3).

Plagiarism Report

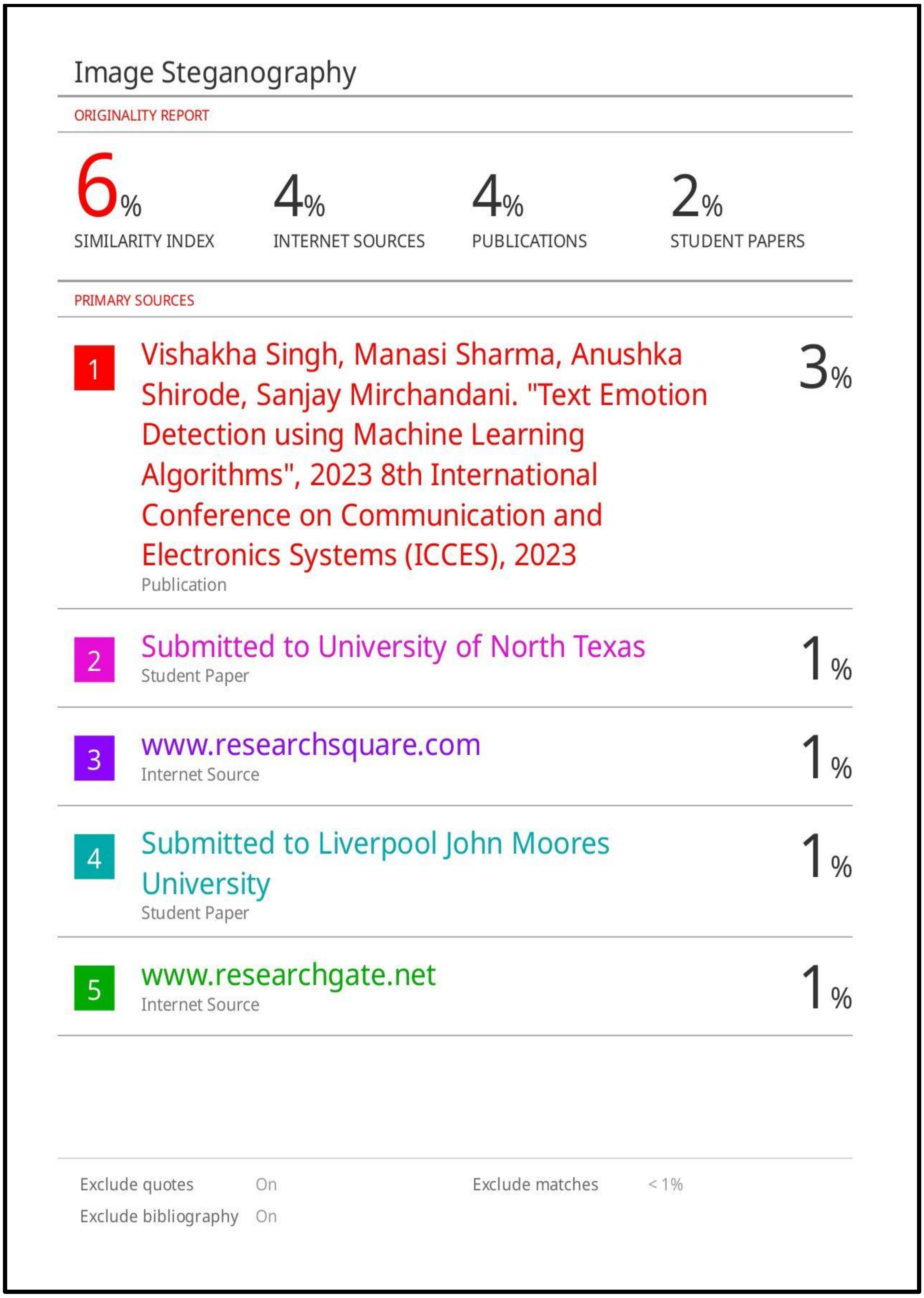


Image Steganography

ORIGINALITY REPORT

6%	4%	4%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

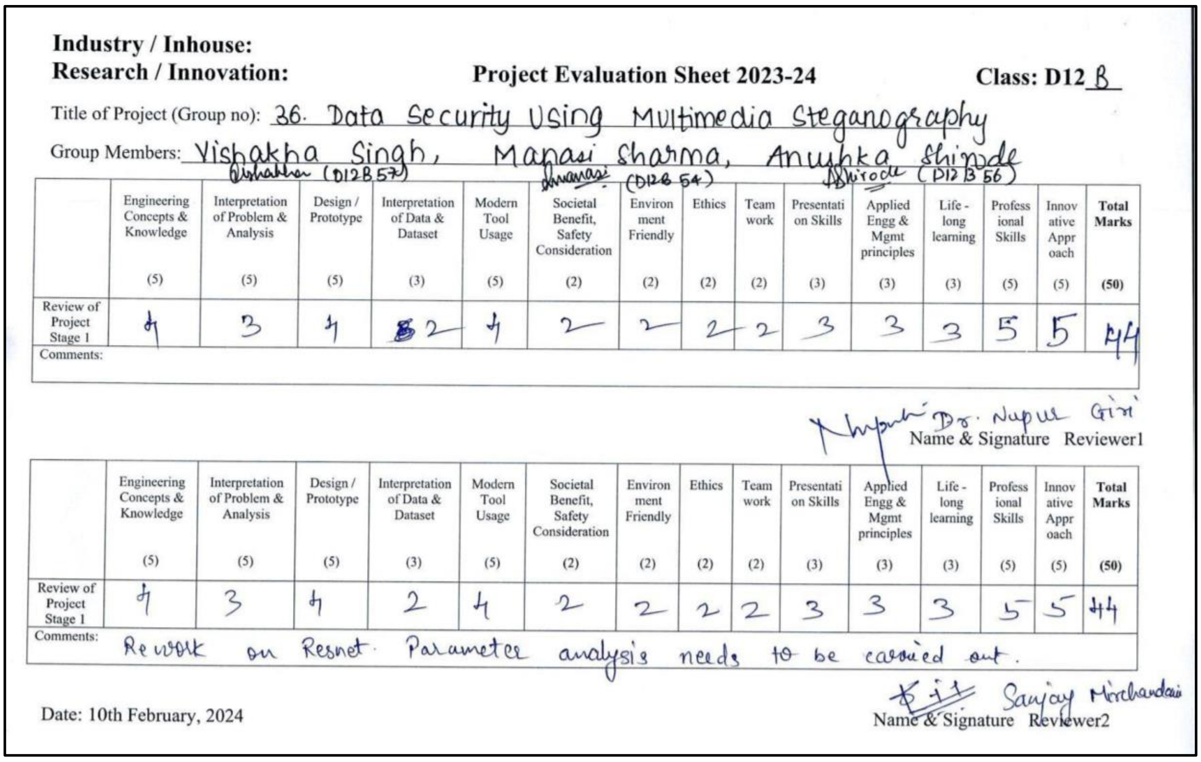
PRIMARY SOURCES

1	Vishakha Singh, Manasi Sharma, Anushka Shirode, Sanjay Mirchandani. "Text Emotion Detection using Machine Learning Algorithms", 2023 8th International Conference on Communication and Electronics Systems (ICCES), 2023 Publication	3%
2	Submitted to University of North Texas Student Paper	1%
3	www.researchsquare.com Internet Source	1%
4	Submitted to Liverpool John Moores University Student Paper	1%
5	www.researchgate.net Internet Source	1%

Exclude quotes On
Exclude bibliography On
Exclude matches < 1%

Project review sheet

Review 1:



Industry / Inhouse:
Research / Innovation:

Project Evaluation Sheet 2023-24 **Class: D12 B**

Title of Project (Group no): 36. Data Security Using Multimedia Steganography

Group Members: Vishakha Singh, Manasi Sharma, Anushka Shinde
[illegible] (D12B57) [illegible] (D12B 54) [illegible] (D12B56)

	Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (3)	Applied Engg & Mgmt principles (3)	Life-long learning (3)	Professional Skills (5)	Innovative Approach (5)	Total Marks (50)
Review of Project Stage 1	4	3	4	2	4	2	2	2	2	3	3	3	5	5	44

Comments:

[illegible] Dr. Nupur Giri
Name & Signature Reviewer1

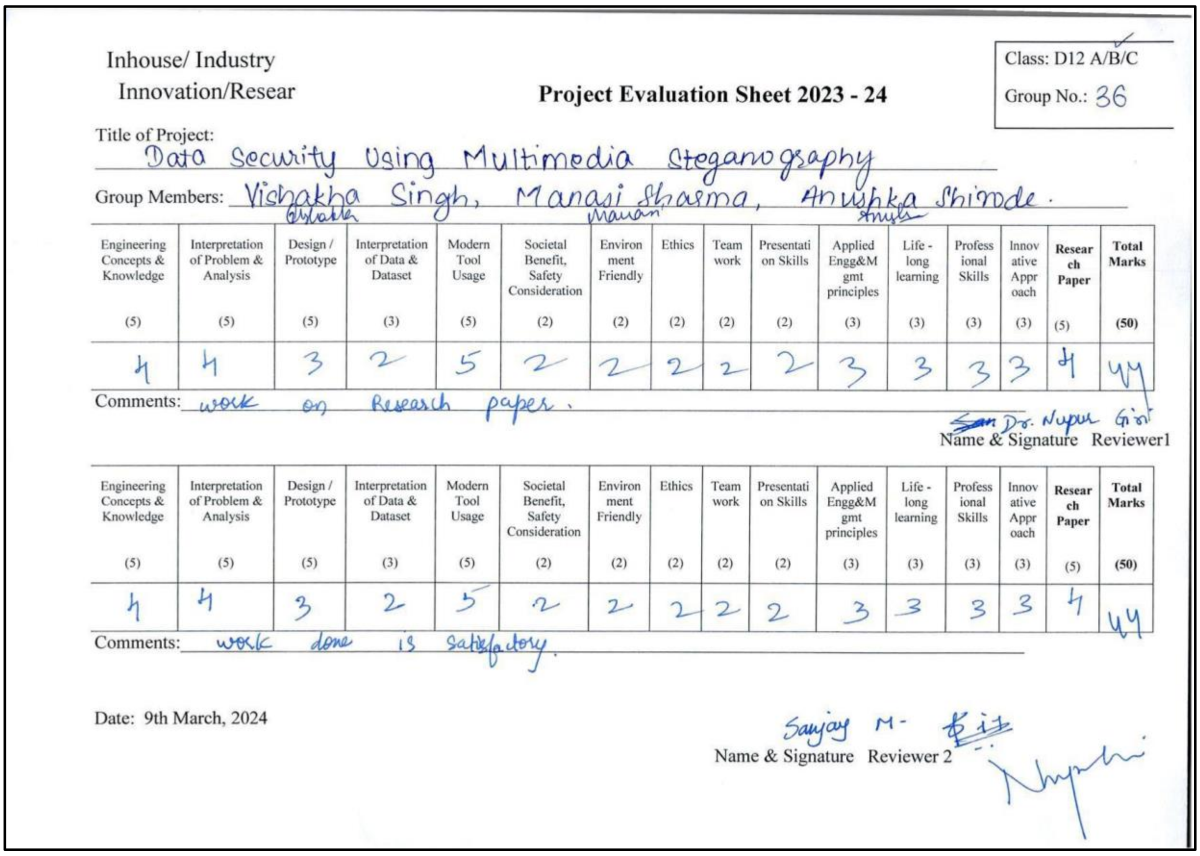
	Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (3)	Applied Engg & Mgmt principles (3)	Life-long learning (3)	Professional Skills (5)	Innovative Approach (5)	Total Marks (50)
Review of Project Stage 1	4	3	4	2	4	2	2	2	2	3	3	3	5	5	44

Comments: Rework on Resnet. Parameter analysis needs to be carried out.

Date: 10th February, 2024

[illegible] Sanjay Mirchandani
Name & Signature Reviewer2

Review 2:



Inhouse/ Industry
Innovation/Resear

Project Evaluation Sheet 2023 - 24

Class: D12 A/B/C
Group No.: 36

Title of Project: Data Security Using Multimedia Steganography

Group Members: Vishakha Singh, Manasi Sharma, Anushka Shinde.

Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (2)	Applied Engg&Mgmt principles (3)	Life-long learning (3)	Professional Skills (3)	Innovative Approach (3)	**Research Paper** (5)	Total Marks (50)
4	4	3	2	5	2	2	2	2	2	3	3	3	3	4	44

Comments: work on Research paper.

[illegible] Dr. Nupur Giri
Name & Signature Reviewer1

Engineering Concepts & Knowledge (5)	Interpretation of Problem & Analysis (5)	Design / Prototype (5)	Interpretation of Data & Dataset (3)	Modern Tool Usage (5)	Societal Benefit, Safety Consideration (2)	Environment Friendly (2)	Ethics (2)	Team work (2)	Presentation Skills (2)	Applied Engg&Mgmt principles (3)	Life-long learning (3)	Professional Skills (3)	Innovative Approach (3)	**Research Paper** (5)	Total Marks (50)
4	4	3	2	5	2	2	2	2	2	3	3	3	3	4	44

Comments: work done is satisfactory.

Date: 9th March, 2024

Sanjay M- [illegible]
Name & Signature Reviewer 2
[illegible]