A Comprehensive Exploration Of Data Security Techniques Using Image Steganography

Vishakha Singh[1], Manasi Sharma[2], Anushka Shirode[3] and Sanjay Mirchandani[4]

[1] V.E.S Institute of Technology, Mumbai-40074, India
2021.vishakha.singh@ves.ac.in

[2] V.E.S Institute of Technology, Mumbai-40074, India
2021.manasi.sharma@ves.ac.in

[3] V.E.S Institute of Technology, Mumbai-40074, India
2021.anushka.shirode@ves.ac.in

[4] 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.

1 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

2 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.

3 Methodology

3.1 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_info
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].

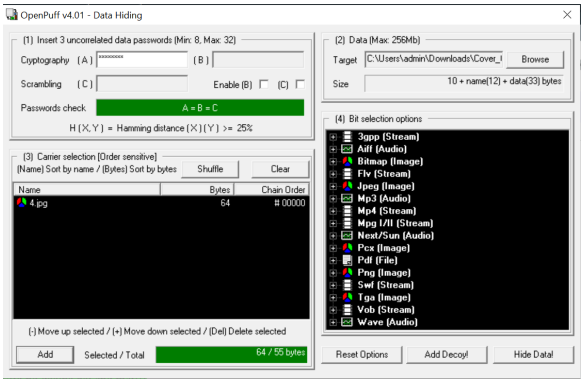

Fig. 2. OpenPuff User Interface

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].

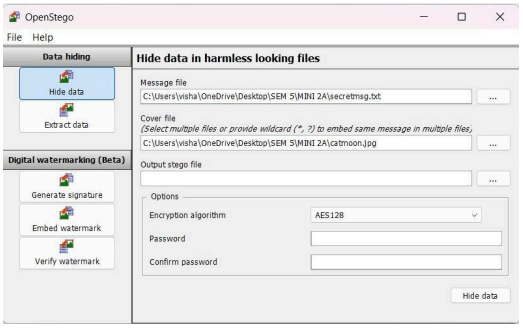

Fig. 3. OpenStego User Interface

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

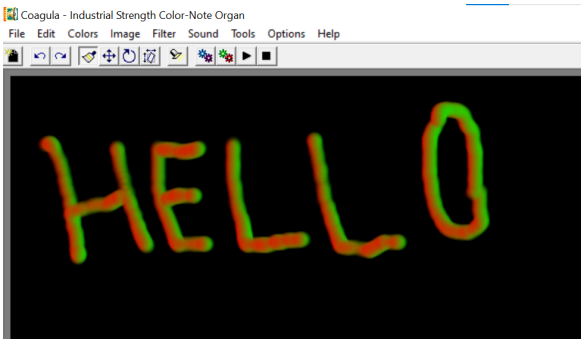

Fig. 4. Coagula User Interface

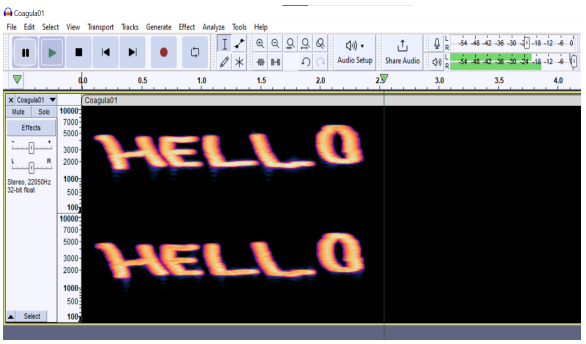

Fig. 5. Audio file as spectrogram

4 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 $\widehat{y}_i$ 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.

5 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 1. 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 2. 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.

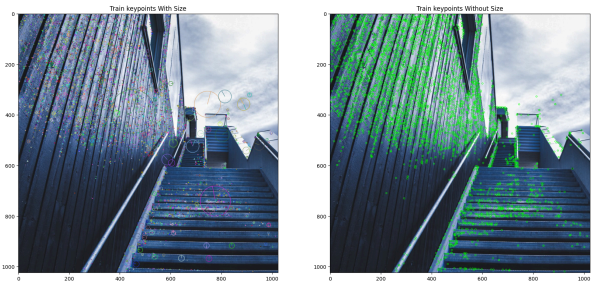

Fig 6. Feature Detection in SIFT

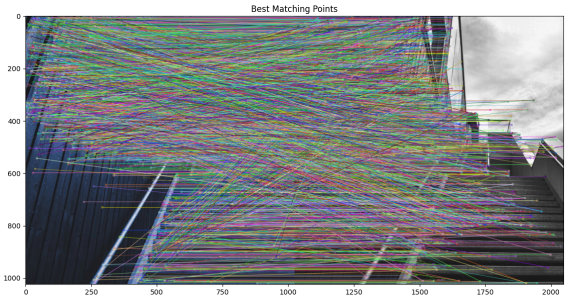

Fig 7. Key Points Detection in SIFT

Table 3. 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, MaxPooling 2D 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,	99.32%	99.29%	0.3016	0.3099
	75:25		99.58%	98.53%	0.0602	0.1004

	70:30	axis=-1, and center=True , scale=True.	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).

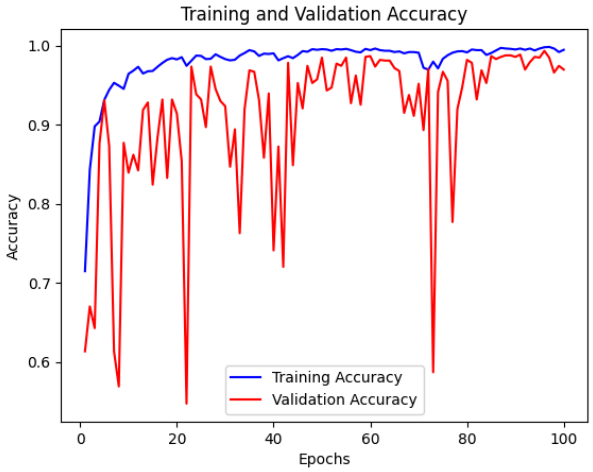

Fig 8. ResNet Accuracy

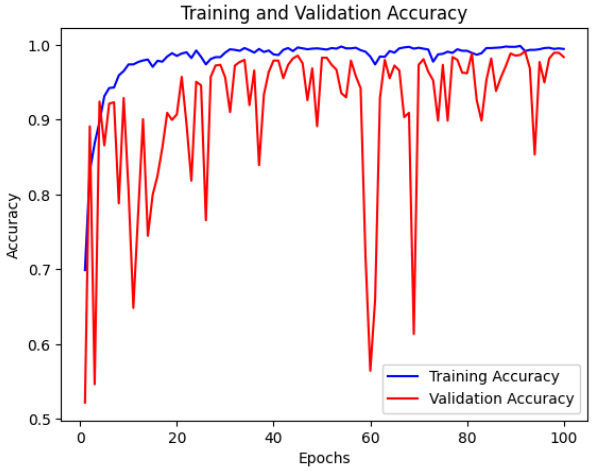

Fig 9. Batch Normalization Accuracy

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.

6 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).