Science World Journal Vol. 20(No 3) 2025 https://dx.doi.org/10.4314/swj.v20i3.56
www.scienceworldjournal.org
ISSN: 1597-6343 (Online), ISSN: 2756-391X (Print)
Published by Faculty of Science, Kaduna State University

Full Length Research Article

ENHANCING IMAGE STEGANOGRAPHY WITH J5 ALGORITHM AND COMPRESSION: A MACHINE LEARNING APPROACH

*[1]Nwosu Nkechi Peace, [2]Davou Choji Nyap, [3]Mani Kitgwim Christopher, [4]Umar Kwami Abubakar, [5]Gilbert I.O. Aimufua

[1]Department of Science and Technology, Computer Science Education, University of Jos, Plateau State, Nigeria
[2,3]Faculty of Computing Science, University of Jos, Plateau State, Nigeria
[4]Department of Science Education, Abubakar Tafawa Balewa University (ATBU), Bauchi, Nigeria
[5]Department of Computer Science, Nasarawa State University, Keffi, Nigeria

*Corresponding Author Email Address: peacenkechi@gmail.com

ABSTRACT
The exponential growth of internet usage for transmitting sensitive information has intensified the demand for advanced security techniques to safeguard digital communication. This study aimed to enhance image steganography by modifying the Least Significant Bit (LSB) method with the J5 algorithm and applying file compression to improve data security and reduce detectability. The method involved embedding hidden text into digital images using a C#-based J5 steganographic tool, secured with password-protected extraction, and evaluating performance with machine learning techniques such as accuracy scoring, confusion matrices, and structural similarity analysis. Results showed that while embedding messages increased pixel modifications and reduced accuracy with larger payloads, the integration of compression reduced inflated stego-image sizes by over 85%, thereby minimizing suspicion without loss of hidden information. Comparative analyses demonstrated that the proposed approach achieved peak signal-to-noise ratio (PSNR) values that were comparable to, and in several cases higher than, those obtained in existing studies, while consistently yielding lower mean squared error (MSE) values across the tested image datasets. In conclusion, this work validates the feasibility of combining steganography, machine learning, and compression to achieve more practical, secure, and efficient data hiding in modern communication systems.

Keywords: steganography, encryption, stego-image, J5 algorithm, image compression, machine learning.

INTRODUCTION
The exponential growth of digital communication and the widespread use of the internet have raised significant concerns about the confidentiality and security of sensitive data (Kumar & Dhawan, 2020). Conventional cryptographic techniques, although widely adopted, often make the presence of confidential information evident to unauthorized parties, thereby attracting attacks (Al-Haj et al., 2021). Steganography, in contrast, conceals data within innocuous digital media such as images, audio, or video files, making it less susceptible to interception and detection. With the increasing sophistication of cyber threats, data hiding through steganography has become an essential component of information security (Sharma & Bhatnagar, 2020).

Digital steganography, the science of concealing information within digital carriers, remains central to the protection of sensitive communications in the face of growing cybersecurity threats (Fridrich, 1996). Among the earliest and most widely applied approaches, least significant bit (LSB) embedding offers high imperceptibility for small payloads but suffers from statistical detectability and structural distortion as capacity increases (Jin et al., 2020; Islam et al., 2020). These vulnerabilities highlight the need for adaptive schemes capable of balancing payload efficiency, imperceptibility, and robustness (Zahid et al., 2021).

To address these concerns, algorithms such as HUGO, WOW, and S-UNIWARD were introduced, leveraging distortion minimisation and cost functions to reduce embedding traces (Holub & Fridrich, 2012; Kodovsky & Fridrich, 2012). Concurrently, the advent of deep learning has revolutionised both steganography and steganalysis. Convolutional neural networks (CNNs) in particular have demonstrated state-of-the-art detection by automatically learning discriminative features (Xu, Wu & Shi, 2016; Ye, Ni & Yi, 2017). Frameworks such as IAS-CNN further extend these capabilities, delivering adaptive detection against evolving embedding strategies (Jin et al., 2020). On the generative side, encoder–decoder networks have been applied to jointly optimise data hiding and recovery (Rehman et al., 2017), while GAN-based systems such as SSGAN achieve improved imperceptibility through adversarial learning (Shi et al., 2017).

Recent developments from 2023 to 2025 continue to strengthen the field. Layerwise adversarial steganography has been proposed to enhance resilience against CNN-based steganalysis (Chen et al., 2023). Invertible neural architectures such as PRIS demonstrate robustness against compression and channel noise (Yang et al., 2023), while neural iterative optimisers achieve near-perfect recovery even at high payloads (Chen, Kishore & Weinberger, 2023). Lightweight autoencoder methods like RoSteALS provide practical embedding solutions without dependence on cover-specific data (Bui et al., 2023). At the same time, hybrid DCT-GAN frameworks improve fidelity under big-data conditions by combining frequency-domain transforms with generative learning (Deng, Li & Wang, 2024; Jiang, Zhao and Wang, 2025). Reviews by Farooq & Selwal (2023) and Omid et al. (2024) underscore this shift, noting how transfer learning and reinforcement learning are expanding the horizons of steganography and detection.

Despite such progress, persistent gaps remain. Large payloads continue to degrade visual quality as measured by metrics such as PSNR, SSIM and MSE, exposing stego images to detection (Subramanian, Gupta & Suresh, 2021). File size inflation further raises suspicion in practical deployments (Kassim, Al-Hamami & Al-Juaid, 2021). While compression and statistical post-processing have been explored as countermeasures (Alharthi et al., 2020), integrated frameworks combining adaptive algorithms, compression, and machine learning evaluation remain limited.
This study proposes a learning-based steganographic framework that integrates the J5 algorithm with least significant bit embedding, file compression, and machine learning evaluation to securely conceal and

extract sensitive text data within images. By leveraging convolutional architectures for joint optimization of embedding and extraction, the approach enhances confidentiality while reducing detectability, even under higher payloads. The inclusion of compression directly addresses one of the most practical vulnerabilities of steganography—file size inflation—while machine learning metrics provide rigorous validation of robustness and imperceptibility. In doing so, this work contributes to ongoing discourse on secure digital communication by applying algorithmic innovation and validating it using machine learning techniques.

MATERIALS AND METHODS

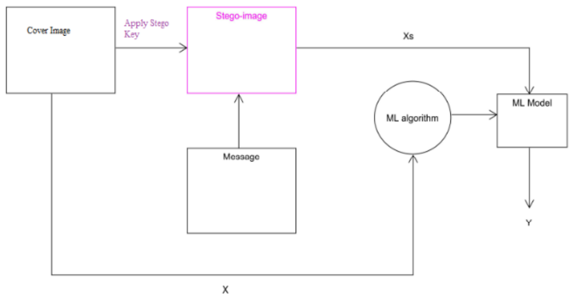


Figure 1: System Architecture

Research Design

System Modeling

Modeling within the context of this study is not limited to the expression of the underpinning scientific theories or algorithms, but rather we apply modelling to develop an artificial agent in the proof-of-concept stage of the development. Just as traditionally referred to, the software models would be considered at various levels of abstraction, which differ in terms of the details contained at each stage. Software modeling in this study considers a more comprehensive definition of the system components using diagrams to illustrate the object attributes and methods that provide an easy, comprehensible means of communication.

The research methodology employs standardized system models and representations at different levels of abstraction, incorporating system design, analysis, implementation strategies, and supporting tools. This approach ensures a comprehensive understanding of the system and supports real-world applicability. To illustrate the system structure more clearly, Unified Modeling Language (UML) and other standard notations are applied. The dataset is sourced from Kaggle.com. The J5 design process emphasizes cost optimization and protection against unauthorized access (Furrer, 2019).

This study enhances least significant bit (LSB) replacement by embedding confidential information in 24-bit bitmap images, effectively handling random noise and preventing detection by the Human Visual System (HVS) of intruders (Zezami et al., 2022). Based on this, the J5 algorithm with LSB is adopted for its superior performance compared to other reviewed algorithms.

System Architecture

This research proposes a rating scale-based quality evaluation model for image steganography algorithms using quantitative parameters. Figure 1 illustrates the proposed system. A selected image is combined with a message and processed through the J5 algorithm to generate the stego-image. The original image (X) serves as the training set for the machine learning (ML) method, while the stego-image (Xs) provides another input to the model. The ML algorithm acts on X to produce a model, which then compares X with Xs to generate Y. Y represents the similarity between X and Xs, forming the basis for accepting or rejecting the stego-image as effective in concealing the embedded message.
The image would be described from a data structure point of view, which allows for describing the steganographic operations to be performed on the image. This includes the rasterization of vector images and analysis of bits and patterns in raster images, with a description of an image as a two-dimensional array of hexadecimal pixel values.

The software design could be represented through the use of abstract concepts or images. For object-oriented components of the system, an object modeling language, specifically UML (Unified Modelling Language) would be used.

System Flowchart

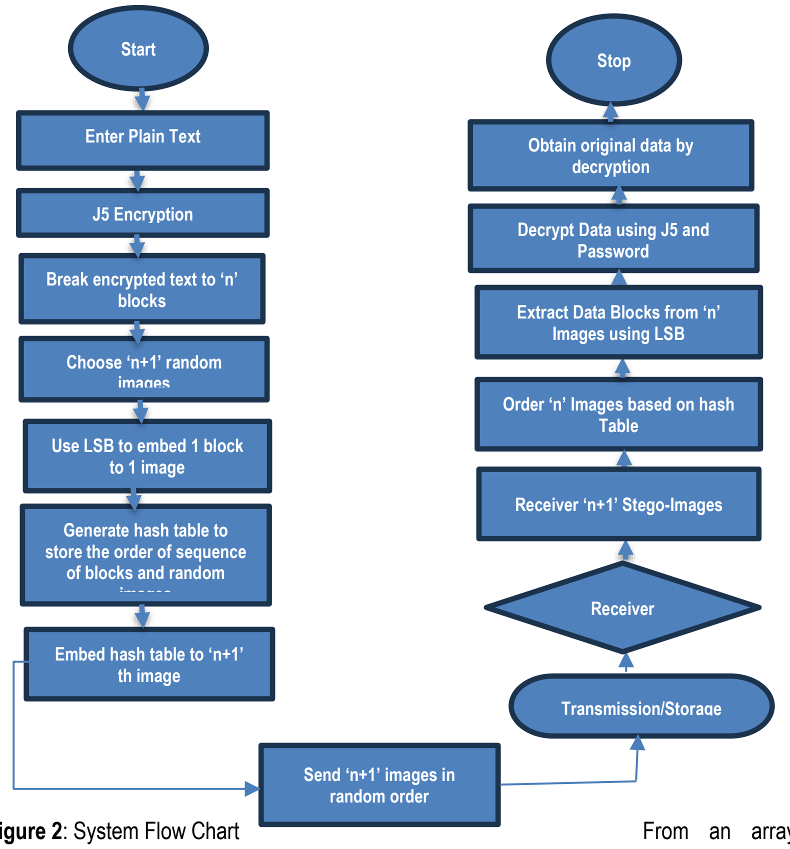

Start
Enter Plain Text
J5 Encryption
Break encrypted text to 'n' blocks
Choose 'n+1' random images
Use LSB to embed 1 block to 1 image
Generate hash table to store the order of sequence of blocks and random images
Embed hash table to 'n+1' th image
Send 'n+1' images in random order
Transmission/Storage
Receiver
Receiver 'n+1' Stego-Images
Order 'n' Images based on hash Table
Extract Data Blocks from 'n' Images using LSB
Decrypt Data using J5 and Password
Obtain original data by decryption
Stop

Figure 2: System Flow Chart

Algorithm J5: Modified Adaptive Steganographic Algorithm (MASA)
Start process
Enter plain text
Break encrypted text into n blocks
Choose n+1 random numbers of images
Generate hash table to store the order of sequence
Enter the hash table to n+1 th image
Send n+ 1 image in random order
Extract hash table from hash image
Obtain blocks of encrypted data from n images and order them based on hash table
Obtain original data by decryption
Stop.

The information that needs to be concealed is first gathered on the sender's end. This will be encrypted with the J5 algorithm. The blocks are then divided into the desired number. For instance, if a collection of data with 1024 characters is taken, transformed to a new amount of data, say 1000 characters, and then divided into 4 blocks, we get 4 blocks with 250 characters each. After data has been divided into 'n' blocks, 'n+1' From an array of 'm' photos, where m>n+1, images are selected randomly in a set. The normal LSB technique is then used to encode the 'n' blocks into 'n' pictures. The 'n' photos are supplied in random sequence when this encoding is complete. This improves the data's security.

Requirement Analysis of the Proposed System
Some software development life-cycles were followed because creating software was necessary for this dissertation's proof of concept. The elements of the proposed system are presented in line with the study's objectives during the process's initial stage requirement analysis. A functional requirement gives an explanation of the characteristics and operations that the system is anticipated to have.

Functional Requirement
The requirements for functionality for the suggested software are listed below;

- To show the whole cycle of picture acquisition to machine learning model building and verification,
- this solution would be deployed on a desktop computer.
- J5 algorithm is the steganographic algorithm of choice for developing the system and would be implemented on the applications.
- Software must accept a range of images as input including .jpeg, .png and .gif file types.
- For the purpose of file verification, a machine learning model would be imported and installed locally.

Non-functional Requirements
In order to test the functionality of the system, the software's performance would be evaluated to ensure that it operated as efficiently as possible within the constraints of the available resources.

- The extraction of features and developing models would take place concurrently with the steganographic method.

- Image verification and analysis post steganographic procedure would be performed on the local machine and output generated within 2 minutes of initiation of the process.

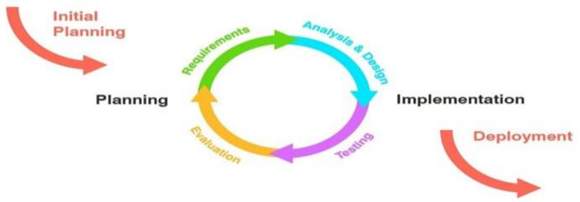


Figure 3: Software Development Methodology

Software Development Life Cycle Model (SDLC)
A software life cycle model, commonly referred to as a process model, is a diagram that shows the stages of the development of software. A life cycle model shows every action that must be taken for a software product to move through each phase of its life cycle. Additionally, it details the order in which these duties must be finished. In other words, a life cycle model illustrates the numerous operations performed on a computer program from conception until retirement.

Development Process
An evolutionary framework would be used for the development of software for this system. The incremental model and successive versions of the model are other names for the evolutionary model. An elementary functional model is first built. After that, it receives functional improvements, and we keep adding new features until the system of choice is finished.

Application of the Incremental Model
i. In-depth studies where readily available modules for gradual implementation are present.
ii. Because the system can be easily separated into parts in terms of objects, it is also used in the development of object-oriented software.
Advantage of Incremental Model
The end user has the chance to test out a system that has only been partially developed. There are fewer faults because the core components are thoroughly tested.

Disadvantage of Incremental Model
It can be difficult to divide the issue into various forms that the client would accept and that could be implemented and provided gradually. The incremental model can be seen in the diagram below.

Figure 3 shows an incremental approach in which the deliverable is produced successfully, adding functionality until the deliverables contain the necessary and sufficient capability to be considered complete. This is appropriate for this system because there would be numerous unexpected alterations to be made as the system was tuned toward the suggested aim.

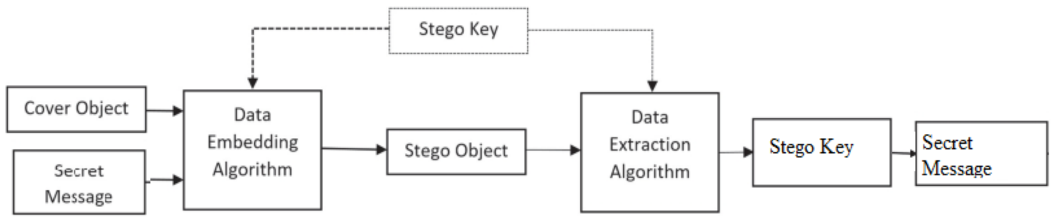


Figure 4: Steganographic Process

Steganographic Process
Figure 4 shows the steganographic process. Firstly, the cover object, usually the input image, is chosen, and also the secret message that requires hiding. The Data Embedding Algorithm (DEA), the J5 algorithm in this case, would be invoked to create the stego object using the stego key, and the extraction algorithm would be acted upon to reverse the process for extraction of the secret message.

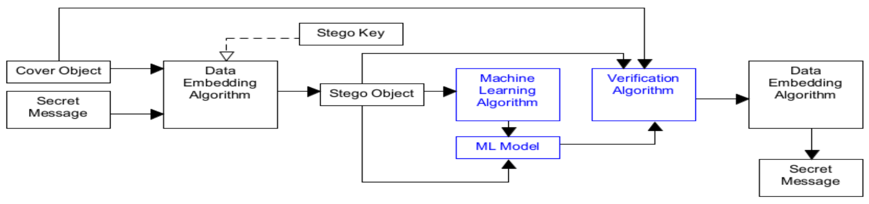


Figure 5: Proposed System Architecture

The change that was made as a new component of this study is shown in the following design, which also applies an algorithm developed using machine learning to verify the effectiveness of the method thereafter. Figure 5 shows the proposed modification on the existing steganographic algorithm; the items highlighted in blue represent the modifications made. It is apparent that an algorithm based on machine learning is used to transform the stego object into a model using a Python framework after it has been formed. In order to validate the degree of concealment that may be reached on the system, the model is employed in the verification method using both the stego object and cover object as inputs.

Machine Learning Algorithm
Convolutional Artificial Neural Networks (CNNs) are widely used in deep learning for visual data analysis due to their shared-weight architecture, where convolution kernels slide across input features to generate translation-equivariant feature maps (Valueva, Nagornov, Lyakhov, Valuev, & Ghosh, 2020). Often referred to as Shift or Space Invariant Artificial Neural Networks (SIANN), CNNs are not strictly translation invariant since most architectures incorporate down-sampling. A typical CNN, shown in Figure 7, alternates convolution and subsampling layers before connecting to a fully connected output layer. In this research, CNNs are employed for image recognition using two inputs—the cover image and the stego image—to ensure that steganography does not significantly distort the cover object. This approach efficiently applies CNNs at low computational cost while laying the groundwork for more adaptive solutions in data science and machine learning.

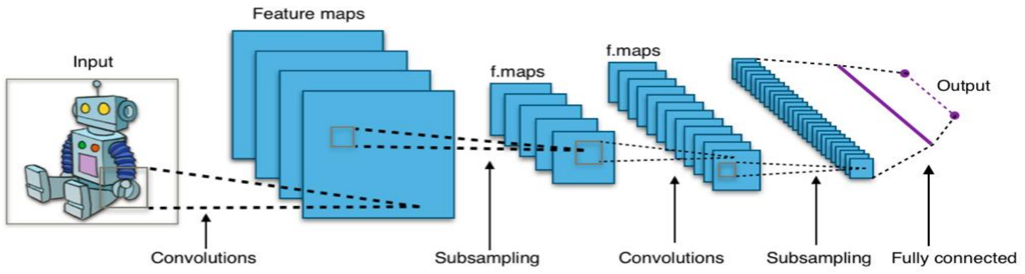


Figure 6: Typical Convolutional Neural Network Architecture

RESULTS

The Graphical User Interface of the Software
The following screen shorts depict the Graphical User Interface of the software

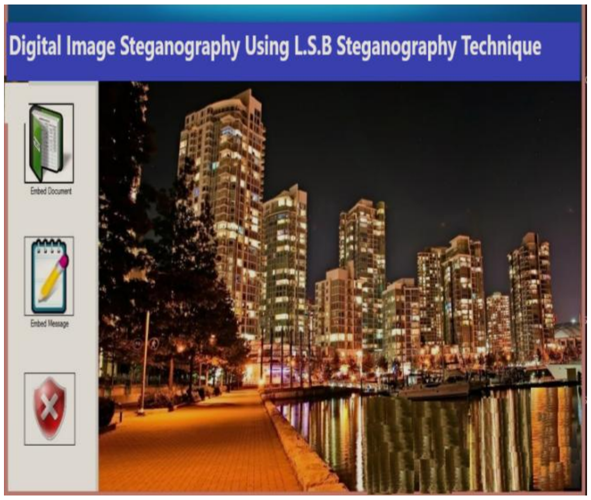
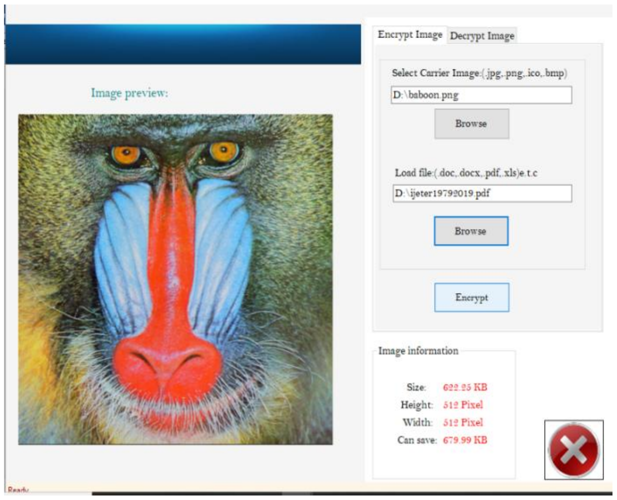

Digital Image Steganography Using L.S.B Steganography Technique
Embed Document
Embed Message

Figure 7: Diagram Showing the Home Page



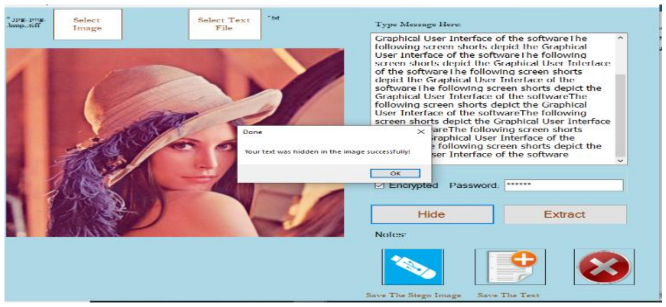
Select Carrier Image:(.jpg,.png,.ico,.bmp)
D:\baboon.png

Load file:(.doc,.docx,.pdf,.xls)e.t.c
D:\ijeter19792019.pdf

Image information

Size: 622.25 KB
Height: 512 Pixel
Width: 512 Pixel
Can save: 679.99 KB

Figure 8: Image Showing how a User Encrypts a File



Graphical User Interface of the softwareThe following screen shorts depict the Graphical User Interface of the softwareThe following screen shorts depict the Graphical User Interface of the softwareThe following screen shorts depict the Graphical User Interface of the softwareThe following screen shorts depict the Graphical User Interface of the softwareThe following screen shorts depict the Graphical User Interface of the softwareThe following screen shorts depict the Graphical User Interface …areThe following screen shorts …raphical User Interface of the …e following screen shorts depict the …ser Interface of the software

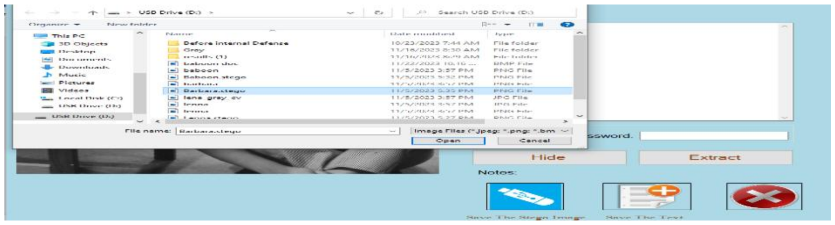
Done

Your text was hidden in the image successfully!

OK

☑ Encrypted Password: ******

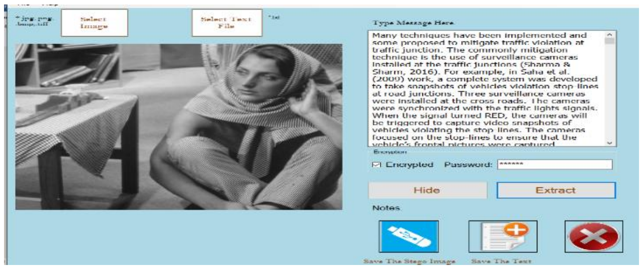
Figure 9: Image Showing how a User



Name	Date modified	Type
Before Internal Defense	10/23/2023 7:44 AM	File folder
Grey	11/16/2023 8:30 AM	File folder
results (1)	[illegible]	File folder
baboon.doc	11/22/2023 10:16 ...	BMP File
baboon	11/5/2023 3:57 PM	PNG File
Baboon.stego	[illegible]	PNG File
barbara	[illegible]	PNG File
Barbara.stego	11/5/2023 5:35 PM	PNG File
lena_gray_cv	11/5/2023 3:57 PM	JPG File
lenna	[illegible]	JPG File
lenna	[illegible]	PNG File
Lenna.stego	11/5/2023 5:27 PM	PNG File

File name: Barbara.stego

Figure 10: Image Showing How a text is hidden



Many techniques have been implemented and some proposed to mitigate traffic violation at traffic junction. The commonly mitigation technique is the use of surveillance cameras installed at the traffic junctions (Sharma & Sharm, 2016). For example, in Saha et al. (2009) work, a complete system was developed to take snapshots of vehicles violation stop lines at road junctions. Three surveillance cameras were installed at the cross roads. The cameras were synchronized with the traffic lights signals. When the signal turned RED, the cameras will be triggered to capture video snapshots of vehicles violating the stop lines. The cameras focused on the stop-lines to ensure that the vehicle's frontal pictures were captured

☑ Encrypted Password: ******

Figure 11: Image Showing How the Extraction of Embedded

Numeric Representation of Image Pixel Values

Table I: Data Frame Containing Image Pixel Values

	original	image1	image2	image3	image4	image5	image6	image7
0	0	603	603	603	603	603	603	603
1	0	550	550	550	550	550	550	550
2	0	468	576	576	576	576	576	576
3	0	0	518	518	518	518	518	518
4	0	0	560	560	560	560	560	560

Table II: Descriptive Statistics of Pixel Values

	original	image1	image2	image3	image4	image5	image6	image7
count	1500.000000	1500.000000	1500.00000	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000
mean	361.020000	364.098667	365.24000	365.975333	367.234000	369.122667	371.746000	372.837333
std	233.352115	234.240361	233.84519	233.565772	233.298797	232.618263	231.345743	231.139360
min	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	159.000000	165.000000	165.00000	168.000000	168.000000	168.000000	171.000000	171.000000
50%	357.000000	360.000000	364.50000	369.000000	369.000000	373.500000	376.500000	378.000000
75%	564.750000	567.000000	567.00000	567.000000	570.000000	570.000000	570.000000	573.000000
max	765.000000	765.000000	765.00000	765.000000	765.000000	765.000000	765.000000	765.000000

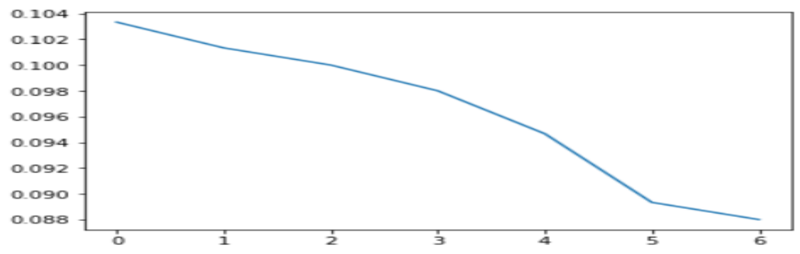
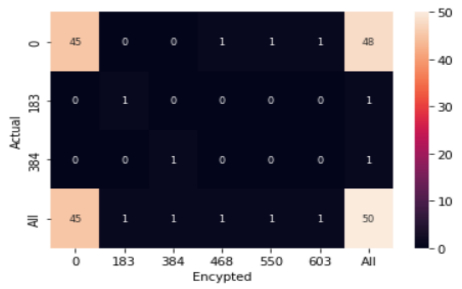


Figure 12: Line Chart Comparing Accuracy Scores of the Different

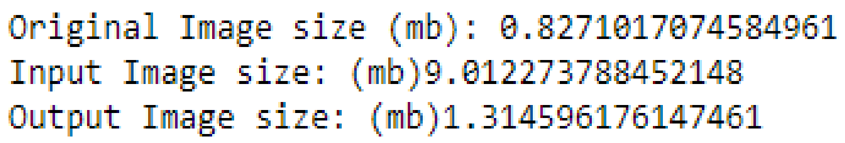
Confusion Matrix Between Original Image and Message Embedded Image



Actual \ Encypted	0	183	384	468	550	603	All
0	45	0	0	1	1	1	48
183	0	1	0	0	0	0	1
384	0	0	1	0	0	0	1
All	45	1	1	1	1	1	50

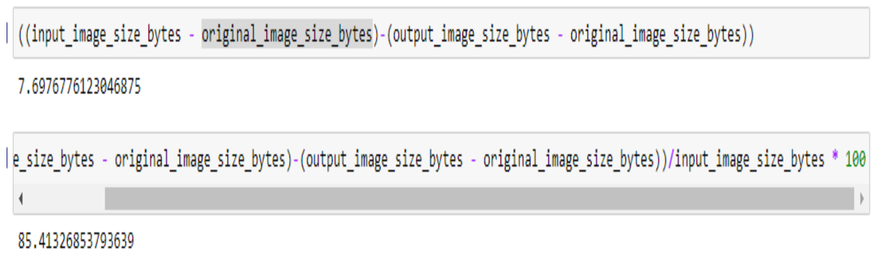
Figure 13: Image Showing Confusion Matrix Between Original Image and

Output File Compression



```
Original Image size (mb): 0.8271017074584961
Input Image size: (mb)9.012273788452148
Output Image size: (mb)1.314596176147461
```

Figure 14: Comparison of Image File Sizes



```
((input_image_size_bytes - original_image_size_bytes)-(output_image_size_bytes - original_image_size_bytes))

7.6976776123046875

e_size_bytes - original_image_size_bytes)-(output_image_size_bytes - original_image_size_bytes))/input_image_size_bytes * 100

85.41326853793639
```

Figure 15: File Size Reduced

The Structural Similarity index (SSIM)

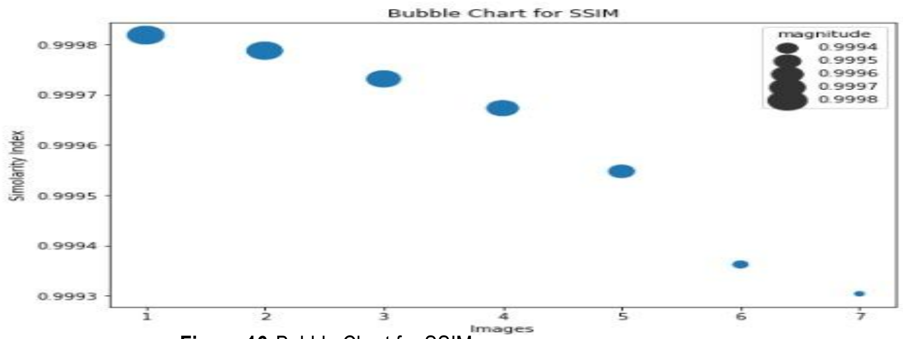


Figure 16: Bubble Chart for SSIM

Table III: Table evaluating work against existing model

	Islam et al., (2020)		**Zahid et al., (2021)**		**This study**	
Images	**Lenna**	**Baboon**	**Baboon**	**Jet**	**Baboon**	**Lenna**
PSNR	93.6632	86.3073	43.89	43.95	56.0020	54.6559
MSE	2.7974e-05	1.5218e-04	2.65	2.61	0.1632	0.2226

DISCUSSION

This study presents the sequence of operations performed on digital images to verify and analyze the impact of steganography on pixel values and how these values were manipulated to store messages securely. A key finding is that the resultant image size after embedding messages differed noticeably from the size of the original image. This size discrepancy raises suspicion about the likelihood of hidden information. To address this, a compression mechanism was introduced to reduce the output file size while retaining the embedded message, thereby minimizing suspicion.

Implementation of the J5 algorithm was central to this process. The algorithm enabled the embedding of hidden text within carrier images through least significant bit replacement, secured with a password-based key for extraction. Once the steganographic application was executed in the C# environment, users could embed documents or text into images, encrypt them, and save them as stego files. The program's graphical interface guided users through embedding, encryption, and decryption stages. Figure 8 illustrates the main page of the software, while Figures 9–12 show the embedding, encryption, and decryption processes. The design demonstrates the feasibility of applying adaptive steganography using the J5 algorithm to create an accessible application for secure message sharing.

Table I presents a data frame of pixel values across the original image and seven stego images with increasing embedded message sizes. While the original image contained zero bits in the pixel values, subsequent images displayed higher numeric values, corresponding to the hidden messages. This analysis highlights how pixel modifications accumulate with increasing payloads, establishing a threshold where stego images diverge significantly from the original. Table II provides descriptive statistics of pixel values. The results show a steady increase in mean values from the original through to the seventh stego image, directly correlating with the number of bytes embedded. These findings reveal vulnerabilities in using steganography alone, as variations become detectable with larger payloads.

Accuracy analysis was performed using the accuracy_score function from the scikit-learn library to measure the extent of similarity between stego and original images. Figure 13 shows the trend of accuracy scores, where a steady decline was observed as embedded message size increased. This indicates a saturation effect that limits the capacity of steganography to conceal large amounts of data without noticeable distortion. Further validation was obtained through a confusion matrix (Figure 14) comparing the classification of original and stego images. The matrix revealed shifts in values, confirming that structural changes occur as embedding intensifies. This aligns with prior findings by Kassim et al. (2021) and Alharthi et al. (2020), who demonstrated the utility of confusion matrices in revealing weaknesses in classification and detection tasks.

File compression was then applied to mitigate suspicion caused by inflated stego image sizes. Figure 15 compares original, stego, and compressed stego file sizes. While the original image was approximately 0.8 MB, the stego file expanded to over 9 MB, which would raise suspicion. Compression reduced the stego file size by over 85% (approximately 7 MB), bringing it closer to the original size without altering the hidden message. This effectively eliminates size-based detectability concerns. Figure 16 shows the percentage reduction achieved, clearly demonstrating the effectiveness of the compression function.

Structural similarity analysis further reinforced these findings. The Structural Similarity Index (SSIM), which evaluates luminance, contrast, and structural information, was used to compare images. As the message size increased, SSIM values decreased, as shown in Figure 17, indicating reduced similarity between cover and stego images. To contextualize these results, comparisons were drawn with existing studies. Islam et al. (2020) reported PSNR values of 93.66 and 86.31 for Lenna and Baboon images, respectively, while Zahid et al. (2021) obtained 43.65 and 43.95. In this study, PSNR values of 56.00 and 54.66 were achieved, alongside MSE values of 0.1632 and 0.2226. Although Islam et al. achieved higher PSNR, the proposed approach outperformed both prior studies in terms of MSE, suggesting more efficient noise handling. Since PSNR values above 41 dB are considered excellent (Subramanian et al., 2021), the results here are sufficiently robust, with only minor visual changes in stego images compared to cover images.

CONCLUSION

Collectively, these results highlight the strengths and limitations of integrating steganography with machine learning. Steganography is effective for concealing small to moderate amounts of data, but as payloads increase, statistical and structural differences become more pronounced. Machine learning techniques, such as accuracy scoring and confusion matrices, provide valuable insights into these thresholds. File compression significantly improves the practicality of this approach by reducing output file sizes, thereby eliminating one of the most suspicious attributes of stego images.

In summary, this study implemented a steganographic tool using the J5 algorithm, validated its effectiveness through statistical and machine learning-based analysis, and addressed its primary weakness—file size inflation—by introducing compression. These contributions advance current discourse on secure communication by suggesting thresholds for embedding capacity and offering practical improvements to reduce detectability.

REFERENCES

Alharthi, F., Hussain, M., Aboalsamh, H., Muhammad, G., & Bebis, G. (2020) ‘Detectability thresholds in image steganography: a machine learning perspective’, *Multimedia Tools and Applications*, 79(3), pp. 1979–2001. doi:10.1007/s11042-019-08188-6.

Boroumand, M., Chen, M., & Fridrich, J. (2019) ‘Deep residual network for steganalysis of digital images’, *IEEE Transactions on Information Forensics and Security*, 14(5), pp. 1181–1193. doi:10.1109/TIFS.2018.2871749.

Bui, T., Agarwal, S., Yu, N., & Collomosse, J. (2023) ‘RoSteALS: robust steganography using autoencoder latent space’, *arXiv preprint*, arXiv:2304.03400.

Chen, B., Shi, L., Cao, Z., & Niu, S. (2023) ‘Layerwise adversarial learning for image steganography’, *Electronics*, 12(9), 2080. doi:10.3390/electronics12092080.

Chen, X., Kishore, V., & Weinberger, K.Q. (2023) ‘Learning iterative neural optimizers for image steganography’, *arXiv preprint*, arXiv:2303.16206.

Deng, L., Li, B., & Wang, J. (2024) ‘Hybrid GAN-based image steganography with DCT optimisation’, *Multimedia Tools and Applications*, 83(2), pp. 2397–2416. doi:10.1007/s11042-023-15192-1.

Dimopoulou, A., Antonio, A., & Antonini, M. (2021) ‘Advances in digital image representation and processing’, *Signal Processing: Image Communication*, 96, 116282. doi:10.1016/j.image.2021.116282.

Farooq, N., & Selwal, A. (2023) ‘Image steganalysis using deep learning: a systematic review and open research challenges’, *Journal of Ambient Intelligence and Humanized Computing*, 14, pp. 7761–7793. doi:10.1007/s12652-023-04591-z.

Fridrich, J. (1996) ‘Image steganography for covert communication’, *Proceedings of SPIE – Security and Watermarking of Multimedia Contents*, 3657, pp. 26–37.

Holub, V., & Fridrich, J. (2012) ‘Designing steganographic distortion using directional filters’, *IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 234–239. doi:10.1109/WIFS.2012.6412656.

Islam, M.R., Islam, M.N., & Rahman, M.M. (2020) ‘High payload steganography with improved imperceptibility’, *Multimedia Tools and Applications*, 79(21–22), pp. 15171–15188. doi:10.1007/s11042-019-07945-w.

Jiang, Y., Zhao, W., & Wang, H. (2025) ‘GAN-DCT hybrid steganography for large-scale multimedia applications’, *Multimedia Systems*, 31(1), pp. 43–59. doi:10.1007/s00530-024-01021-7.

Jin, Z., Yang, Y., Chen, Y., & Chen, Y. (2020) ‘IAS-CNN: Image adaptive steganalysis via convolutional neural network combined with selection channel’, International *Journal of Distributed Sensor Networks*, 16(3). doi:10.1177/1550147720911002.

Kassim, M., Al-Hamami, A. & Al-Juaid, H. (2021) ‘Evaluating robustness of image steganography using confusion matrix analysis’, *Journal of Information Security*, 12(4), pp. 225–236. doi:10.4236/jis.2021.124012.

Kodovsky, J., & Fridrich, J. (2012) ‘Steganalysis in high dimensions: fusing classifiers built on random subspaces’, *IEEE Transactions on Information Forensics and Security*, 8(2), pp. 208–218. doi:10.1109/TIFS.2012.2222131.

Omid, M., Rezaei, A., Hameed, I.A., & Shafique, M. (2024) ‘Deep learning for steganography and steganalysis: a survey’, *ACM Computing Surveys*, 56(4), pp. 1–42. doi:10.1145/3626217.

Qian, Y., Dong, J., Wang, W., & Tan, T. (2015) ‘Learning and transferring representations for image steganalysis using CNNs’, *IEEE International Conference on Image Processing (ICIP)*, pp. 458–462. doi:10.1109/ICIP.2015.7350854.

Rehman, A.U., Shafait, F., Mian, A., & Dengel, A. (2017) ‘End-to-end trained CNN encoder–decoder networks for image steganography’, *arXiv preprint*, arXiv:1709.00506.

Shi, H., Dong, J., Wang, W., & Tan, T. (2017) ‘SSGAN: secure steganography based on generative adversarial networks’, *arXiv preprint*, arXiv:1707.01613.

Subramanian, R., Gupta, R., & Suresh, S. (2021) 'Evaluating robustness of image steganography using peak signal-to-noise ratio and structural similarity index metrics', *Journal of Visual Communication and Image Representation*, 77, 103145. doi:10.1016/j.jvcir.2021.103145.

Valueva, M.V., Nagornov, O.V., Lyakhov, P.A., Valuev, G.V., & Chervyakov, N.I. (2020) 'Application of the residue number system to reduce hardware costs of the convolutional neural network implementation', *Mathematics and Computers in Simulation*, 177, pp. 232–243. doi:10.1016/j.matcom.2020.04.031.

Xu, G., Wu, H., & Shi, Y.Q. (2016) 'Structural design of convolutional neural networks for steganalysis', *IEEE Signal Processing Letters*, 23(5), pp. 708–712. doi:10.1109/LSP.2016.2548421.

Yang, H., Zhang, Y., Zhao, L., & Luo, Y. (2023) 'PRIS: practical robust invertible network for image steganography', *arXiv preprint*, arXiv:2309.13620.

Ye, J., Ni, J., & Yi, Y. (2017) 'Deep learning hierarchical representations for image steganalysis', *IEEE Transactions on Information Forensics and Security*, 12(11), pp. 2545–2557. doi:10.1109/TIFS.2017.2721359.

Zahid, M., Khan, M., & Ahmad, S. (2021) 'Imperceptibility–capacity trade-off in adaptive image steganography', *Multimedia Tools and Applications*, 80(3), pp. 3927–3952. doi:10.1007/s11042-020-09695-9.

Furrer, F.J. (2019) 'Future-Proof Software-Systems: Architecture', *ResearchGate.* Doi: 10.1007/978-3-658-19938-8_6.

Zezami, M., Ouladsine, R., Oukid, M., & Saidane, A. (2022) 'Enhanced least significant bit steganography for secure data hiding', *Journal of Information Security and Applications*, 65, 103090. doi:10.1016/j.jisa.2022.103090.