WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

An Improved Image Steganography System Using Convolutional Neural Networks

Ofoegbu, C. I. [a*], Nwokorie, E. C. [a], Onyemauche, U.C. [a], Okolie S. A [a], Amadi, C.O [a]

[a] *Department of Computer Science, Federal University of Technology, Owerri, Nigeria*

A B S T R A C T

This study introduces an innovative image steganography method that combines convolutional autoencoders with the ResNet architecture, evaluated using the CIFAR dataset. It overcomes traditional steganography limitations by leveraging deep learning, achieving high imperceptibility, low computational power needed and substantial hiding capacity. The approach involves a preprocessing model for feature extraction and an operational model for embedding and extraction, both based on ResNet. Quantitative comparisons with previous methods demonstrate superior performance, with PSNR and SSIM exceeding 30 dB and 0.98. This method enables concealing color images within others, offering significant capacity

Keywords: ***Image steganography, Deep neural networks, Autoencoder, ResNet, CIFAR dataset***

1. Introduction

Steganography is the technique of hiding secret data within a cover medium, such as an image, audio, or video file, without altering the perceptual quality of the medium to avoid detection. Image steganography is the art and science of hiding secret information within digital images without degrading their quality or visual appearance. This method is commonly used for secure communication and data storage, as well as for digital watermarking and fingerprinting. The secret data is then extracted once it arrives at its destination. Steganography, in conjunction with encryption, can be used to further conceal or safeguard data. With the advancement of digital technology and the increasing importance of data privacy and security, the goal of steganography is to ensure the security and confidentiality of information by embedding it onto a cover medium in such a way it remains hidden from unauthorized parties (Sana et.al., 2021)

Over the years, various Image steganographic techniques have been developed, ranging from simple methods based on LSB (Least Significant Bit) substitution to more advanced techniques based on frequency domain analysis and deep learning (Fridrick et. al., 2001). One of the recent advancements in image steganography is the use of Convolutional Neural Networks (CNN) for image embedding and extraction. The use of CNNs in image steganography has several advantages. Firstly, CNNs can learn complex and non-linear relationships between pixels in images, making them more effective at detecting and hiding secret information.CNN is a type of deep learning neural network architecture that has been widely used in various computer vision tasks, such as image recognition/object detection, classification, and segmentation tasks. In the context of image steganography, CNN is used to embed secret data within an image by modifying the pixel values. The use of CNN in image steganography has shown promising results in terms of security and robustness. CNNs have shown promise in image steganography due to their ability to learn and represent complex patterns in data, and their robustness against various attacks.

One of the challenges of using CNNs for image steganography is to ensure the trade-off between the level of secret message embedding and the image quality. It is important to embed the maximum amount of secret information while minimizing the distortion introduced in the host image.One of the earliest works on using CNN for image steganography was proposed by (Yang et. al., 2019), where they used a CNN-based model to embed and extract secret information in the frequency domain. They showed that the CNN-based steganography approach outperformed traditional image steganography methods in terms of security and capacity. Another study by Zuo *et al*. (2020) proposed an optimized steganography method based on deep convolutional autoencoder (CAE) architecture. The method uses a CAE-based encoder to embed secret data in the image and a decoder to extract the secret data. The results showed that the proposed method achieved higher hiding capacity and better visual quality than existing steganography methods. Xu et al. (2018) proposed a steganography method using a CNN with a multi-scale structure to embed secret data into the image. Also, Wu *et al.* (2019) proposed a steganography method using a deep residual network (ResNet) to embed secret data into images. The ResNet architecture allowed for the embedding of secret data with high accuracy while maintaining image quality. In recent years, several other studies have also explored the use of CNN-based models for image steganography, such as the steganography method using dual-path convolutional neural network (DP-CNN) proposed by Zhang *et al*. (2021). The DP-CNN model uses dual-path architecture to embed and extract secret information in the spatial domain. The results showed that the proposed method achieved higher hiding capacity and better security than existing steganography methods.

Moreover, Zuo *et al*. (2020) proposed an optimized steganography method based on deep convolutional autoencoder using CNNs. The method utilized an autoencoder architecture to compress the secret data and then embed it into the image. The proposed method achieved high embedding capacity and improved the security of steganography.

The use of CNNs in image steganography has shown promising results in achieving high embedding capacity and maintaining image quality. As the need for data privacy and security continue to grow, it becomes a burden on the research community to manage this growth by further research in this area that can lead to the development of more secure and efficient steganography.

There are several problems and limitations associated with the use of image steganography, some of which have been highlighted in the literature.

Several approaches for image steganography have been developed in recent studies to address some of these issues. Although each approach still suffer from some limitations The conventional image steganography techniques suffer from high computational power, low embedding capacity and poor visual quality of the stego images and the capacity of the hidden information. As the amount of information to be hidden increases, the quality of the cover image may be significantly degraded, which could raise suspicion and compromise the effectiveness of the steganographic technique (Chen , 2018). However, the existing approaches using CNN have some limitations. For instance, in the work of (Zhang et al., 2020), they used a CNN model to embed secret messages into images, but the method had limitations in terms of the image quality after embedding the secret message. Similarly, in the work of (Han et al., 2019), they used an autoencoder model for image steganography, but the method was limited in terms of embedding capacity and the robustness of the hidden messages against image processing attacks. Another challenge is the potential vulnerability of the image steganographic system to attacks aimed at detecting and extracting the hidden information and also reducing computational power needed to run on lighter devices. These attacks may involve statistical analysis, machine learning algorithms, or deep learning techniques that could effectively reveal the presence of the hidden information and its content (Zhang et al., 2020). Researchers have conducted various studies and reviews on image steganography using CNN trends and models, and have shown that gaps still exists in this sub-domain.

Although CNN have shown promising results in increasing the embedding capacity and maintaining the visual quality of stego images, they still have limitations in terms of hiding large amounts of data while minimizing distortion.

This study therefore, attempts to improve an existing CNN-based image steganopraphic system in terms of addressing gaps and inefficiencies in its computational complexities and image encryption to produce better data security. As yet, there has been no much independent study conducted on the chosen existing system that is available online. It is therefore on this premise that this study becomes imperative to make contribution to the constant attempt to improve CNN-based image steganographic systems driving towards achieving a system that can effectively hide large amounts of data in an image with minimal distortion and high visual quality. The aim of the study is to design an improved image steganography system that uses convolution neural networks (CNN). To achieve this main objectives, to develop an image steganography model based on image embedding and extraction using CNN; to implement the model developed using python programming; to evaluate the prototype using suitable evaluation methods.

2. Analysis of the Existing System

An existing study model was chosen for this study. The entire system consists of 3 CNNs: The Preprocess Network. This network prepares the secret image to be hidden the purpose is to transform the color-based pixels to more useful features so that in can be encoded such as edges. It contains 50 filters of (3X3, 4X4, 5X5) patches there are total 6 layers of this kind. . The Hiding Network: This network will take the output of the preparation network and will then create a Container Image. It is a CNN with 5 convolutional layers that have 50 filters of (3X3, 4X4, 5X5) patches. This layer consists of total of 15 convolutional layers in this network. The Reveal Network: Converts the Container image to the original image this network is used for decoding. Which has 5 convolutional layers that have 50 filters of (3X3, 4X4, 5X5) patches. This layer consists of total of 15 convolutional layers in this network, as show in Fig. 1..

As a way of improving the range of protecting privacy and data across the internet and between direct communications, an image steganography model was developed to assist users with data protection. While there exist several image steganographic models and systems using deep learning, majority of them either lack robustness or are proprietary and not readily available for use, which also brings about the difficulty in gaining access to their internal framework and conceptual design. These image steganographic systems are also on ‘pay and get’ basis and in many instances very expensive to purchase [9]. This formed the basis for choosing the selected existing image steganography model as a study model. After reviewing the selected existing system, we found need for improvement of the system most especially in its handling encoding and decoding of secret images and cover images.

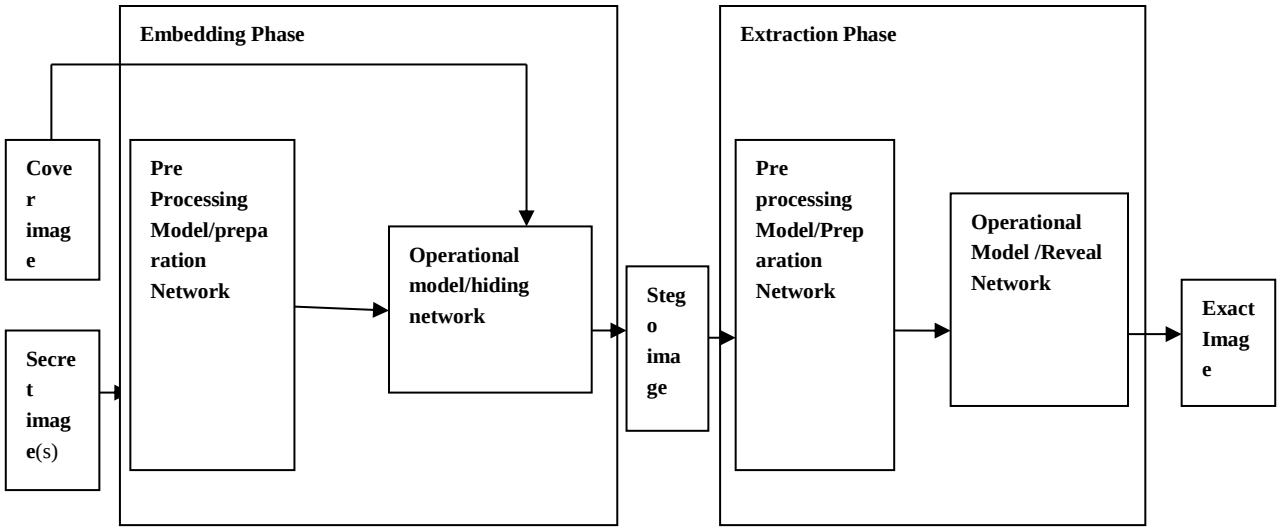

Fig. 1. General Structure of the Existing System (Sasi el al., 2020)

2.1 Features of the Existing System

Here are some of the key features of the existing system: the image steganography system described here utilizes CNN (Convolutional Neural Network) model architecture. Let's break down its features and understand what each component does:

1. Prep Networks:

Structure: Consist of two layers, each containing three independent Conv2D layers.

- Layer Details: The Conv2D layers within each layer have channel configurations of 50, 10, and 5. they employ kernel sizes of 3, 4, and 5, respectively.
- Stride Length: The stride length along both axes is maintained at one to preserve the output image's dimensions.
- Padding: Appropriate padding is applied to each Conv2D layer to ensure that the output image retains the same dimensions.
- Activation: After each Conv2D layer, a Rectified Linear Unit (ReLU) activation function is applied.

2. Concealing Network:

Structure: Comprises three layers, and each layer consists of three individual Conv2D layers.

- Layer Details: The Conv2D layers in this network have a structure similar to the Conv2D layers in the Prep Network.

3. Reveal Network:

Structure: Resembles the hidden network, with three levels of Conv2D layers having a comparable structure.

2.2 Algorithm for the Existing Model

Description: The implemented system employs a CNN for concealing an image within another image, rendering the secret image effectively invisible to observers. This approach is more efficient than traditional LSB (Least Significant Bit) manipulation.

- Encoding Process: Neural networks determine where in the cover image to hide information. This process minimies noticeable changes to the container image.
- Decoding Process: A decoder network is trained to reveal the secret image from the container image without significantly altering the container image.
- Channel Usage: The secret image is effectively hidden in all three color channels of the container image.
- Security Assumption: It assumes that the intruder does not have access to the original image.
- Activation Functions: The system has been tested with three different activation functions: Rectified Linear Unit (ReLU), Tanh, and Scaled Exponential Linear Unit (SELU).
- ReLU: A widely used activation function in CNNs.

- Tanh: Another activation function that squashes output to the range [-1, 1].
- SELU: Used to address the vanishing gradient problem associated with the ReLU activation function.

2.3 Limitations of the Existing System

a) One major challenge is the potential for detection by advanced steganalysis techniques. While CNN can provide a high level of security, they are not immune to detection. Adversarial attacks, in which an attacker attempts to identify hidden information, can be developed to counter steganography using CNN s. This highlights the importance of ongoing research to develop new and more robust steganography techniques (Wei et. el., 2020).

b) Another limitation of the existing system is the computational resources required for training and testing. These techniques involve complex neural network models that require large amounts of data and processing power to train effectively. As a result, the development of steganography systems using CNN can be resource-intensive and time-consuming.

c) Potential for Detection: While the system is designed to provide a high level of security, it is not immune to detection. Advanced steganalysis techniques can potentially identify hidden information, especially if the attacker has knowledge of the steganography technique used.

d) Limited Robustness: The system may not be robust to various attacks, such as image tampering or noise addition, that could degrade the quality of the stego-image and make it easier for an attacker to detect the hidden information.

e) Limited Embedding Capacity: While the system can embed a large amount of secret information in an image, the capacity of the system is still limited. This means that there may be a tradeoff between the amount of information that can be hidden and the quality of the stego-image.

f) **Model Complexity**: The multi-layered CNN architecture, while effective for steganography, introduces complexity. Understanding, training, and fine-tuning such models can be challenging for users with limited experience in deep learning.

g) **High Computational Cost**: The described CNN architecture, with multiple layers and Conv2D operations, can be computationally intensive. Training and running such models, especially with large datasets, may require substantial processing power.

h) **Resource-Intensive**: Implementing this system on resource-constrained devices, such as mobile phones or IoT devices, might be challenging due to its computational demands.

i) **Loss of Image Quality**: The process of hiding one image within another can lead to a loss of image quality. Depending on the complexity of the network and the degree of hiding, the stego image may exhibit artifacts or imperfections that weren't present in the original cover image.

j) **Visibility of Changes**: Even though efforts are made to minimize noticeable changes to the container image, there may still be cases where alterations are discernible to the human eye, especially if a high level of data is hidden.

3. Proposed System

An improved system is proposed to achieve better image quality and data security. The proposed model is a stand-alone application. The addition of ResNet allows for better feature extraction, which in turn leads to better quality stegoimages. The combination of convolutional autoencoders based of ResNet architecture helps to overcome some of the security and image quality flaws that are present in autoencoder-based image steganography systems/models. The system will be able to embed a secret message in image without significantly degrading the quality of the image. The core of an image steganography system is in the pre-processing phase, which involves using techniques such as resizing, normalization and blurring during image during pre-processing. Hence, the proposed model provides an efficient algorithm that can effectively handle these tasks and deliver a more reliable summary. A major advantage of the proposed system is its ability to improve the security of the hidden message. By combining ResNet and autoencoders, the system makes it more difficult for an adversary to detect the hidden message various techniques. This is because the ResNet extracts more robust features from the image, which makes it more difficult for adversary to distingish between the original image and stego image.

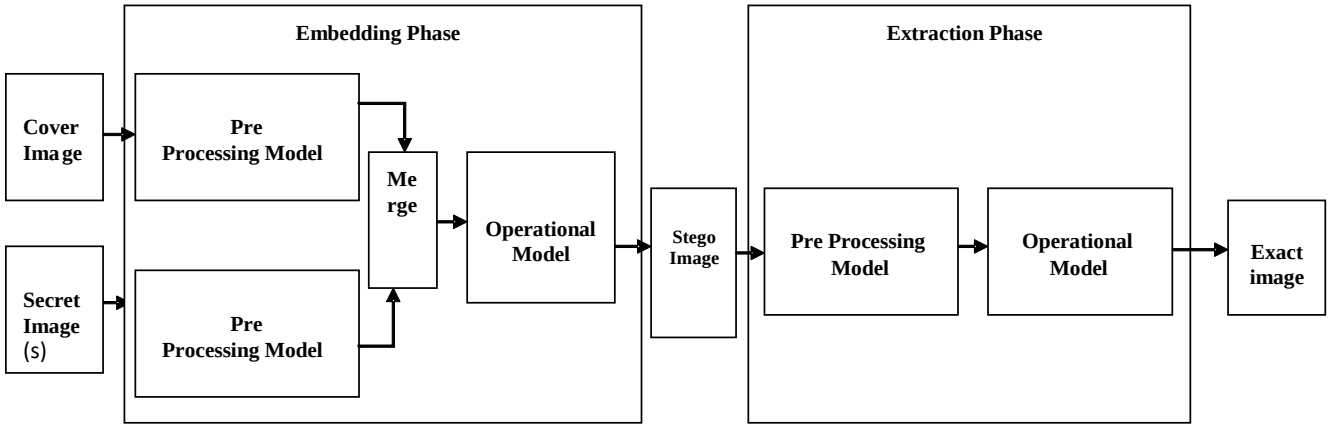

Fig. 2 Preprocess model

As shown in the proposed system model architecture, there are two deep neural networks in this scheme. These models are trained simultaneously at each phase. Although they act as a unified network, it is easier to explain their operations and properties separately. The first model, i.e., the preprocess model prepares images to be ready for the second model. The second model that is the operational model, takes the outputs of the preprocess model as inputs to produce an image (stego or extracted image).

These two models work in two phases: the embedding and extraction phases. In the embedding phase, secret and cover images are passed through the preprocessing model. Then, they are merged and a feather map with 128 channels is obtained. Next, this feather map is passed through the operational model to generate the stego image. The extraction phase recovers the secret image from the container stego image. During the second phase, stego image is passed through preprocess and operational models with the aim of extracting the secret image that is hidden in the stego image. Let c, s be the cover and secret images, respectively. Producing a stego image h is the main goal of the embedding phase where h is similar to s, and, the main aim in the extraction phase is to recover the extracted image e which is similar to s. Mathematically, the proposed scheme can be expressed as follows. In embedding phase, s' and c' are feather maps with 64 channels that are generated by the preprocess model ($Prep$).

$$s' = Prep(s)\ ,\ c' = Prep(c) \quad (1)$$

Then, the operational model (Om) produces the stego image h.

$$h = Om(s'\ ,\ c') \quad (2)$$

During the extraction phase, e recovers from h. In other words,

$$e = Om(Prep(h)) \quad (3)$$

More details on each of the models are given in the features of the proposed system

Generally, images contain redundant data, and the burden on the embedding and extraction phases will be reduced by extracting the most meaningful features. So, in the embedding and extraction phases, the proposed scheme extracts the primary information from the secret, cover, and stego images as feature maps and prepares them for the operational model. In the both embedding and extraction phases, features produced by preprocessing model are used instead of raw form of images for steganography. The input size is $M \times$, which represents the width, height and number of channels of the image. This model contains three convolutional layers with increasing number of filters.

The overall block-diagram of the proposed image steganography scheme. are as follows. The kernel is of the size 3×3, the size of stride is 2, and the padding size 1. For initial filters, we should use fewer number of filters to extract lower-level local features, for example the edges of the images. By increasing the number of filters after each layer, the model can find more sophisticated features. There are three convolutional layers in each phase. The number of filters in the first, second and third convolutional layer in the embedding phase is 16, 32, and 64 and in the extraction phase, is 32, 64 and 128, respectively. After each convolution layer, a ReLU and a Batch Normalization (BN) layer is used. The reason of using the ReLU layer is increasing the nonlinear fitting property of the model. As a result, it will learn the details of the secret image, cover, and stego images better. Also, the batch normalization layer is used (except for the first and last convolutional layers) to speed up the network. This model is like an encoder in an autoencoder that produces a feature map from an image. B. Operational model Similar to the preprocess model, the operational model is used in both the embedding and extraction phases. In the embedding phase, the two images passed through the preprocess model are merged and a feature map with 128 channels is generated.

Also, the feature map that is produced in the preprocessing model of the extraction phase has 128 channels. So, the input tensor of the operational model in both the embedding and extraction phases has 128 channels. It is obvious that the output size is always $M \times N \times O$. In the embedding phase, this model generates a stego image (s) and during the extraction phase, this model generates the extracted image (e). Convolutional neural networks are one of the best-known architectures for autoencoders in encoder and decoder. But when the depth of CNN architecture increases, the problem of vanishing descent arises which affects the accuracy. In this paper, the aforementioned challenge is addressed using the residual block connection technique which is known as ResNet. Utilizing this connection method, the rate of reusing the features is increased. Consequently, the imperceptibility of the stego images is

enhanced, and finally, in the extraction phase, the quality of extracted images is improved. In other words, the similarity between the cover and stego images and also between the secret and extracted images will be increased. Fig. 2 presents the idea of feature connection in ResNet. The symbol "⊕" in this figure, means the element-level addition that is known as shortcut. It should be noticed that the two feature layers (x) and x must have the same size. The idea of cross-layer connection has been applied in Resnet and nowadays, and we have witnessed widespread applications of ResNet as a popular classification model.

The operational model is contained three residual block connections, including 2 convolution transpose operations, Batch normalization (BN) and Leaky-ReLU layers. These layers work like a decoder in an autoencoder and reproduce an image from a feature map. The detailed characteristics of the convolution transpose operations are as follows. The kernel is of the size 3×3 and 4×4, the size of stride is 2 and 1, and the padding size 1.

In summary, the architecture of the image steganography system consists of two main components: the preprocessing model and the operational model. It's designed to extract meaningful features from secret, cover, and stego images and to embed and extract secret information effectively. The operational model has been simplified by reducing the convolutional layers to two while maintaining feature-rich connections through residual blocks (ResNet-like).

3.1 Feature of the Proposed System

The proposed system was designed based on two machine learning techniques.

- Better image quality: The system is able to embed a secret message in an image without significantly degrading the quality of the image.
- Versality: The system allows for the use of different image formats such as PNG and JPG, which makes it more versatile.
- Combined latent representation: the system uses a combined latent representation of Resnet50 encoder model and vae decoder model for improved encoding and decoding operations.
- Efficient feature extraction: The use of ResNet allows for better feature extraction, which in turn leads to better stego images.
- At the image and security level, the algorithm of the proposed system performs image feature extraction and compression efficiently.
- The proposed system implements an algorithm dynamic enough to manage variations in image patterns.
- The proposed system is a stand-alone application and will not require the availability of internet connection to function.
- User-friendly interface: The system has a user-friendly interface that allows for easy interaction with the system.
- It integrates the display of relevant statistics for the original image as well as the generated stego image. Statistics such as the size of the image, the resolution, the file format, and the colour depth.

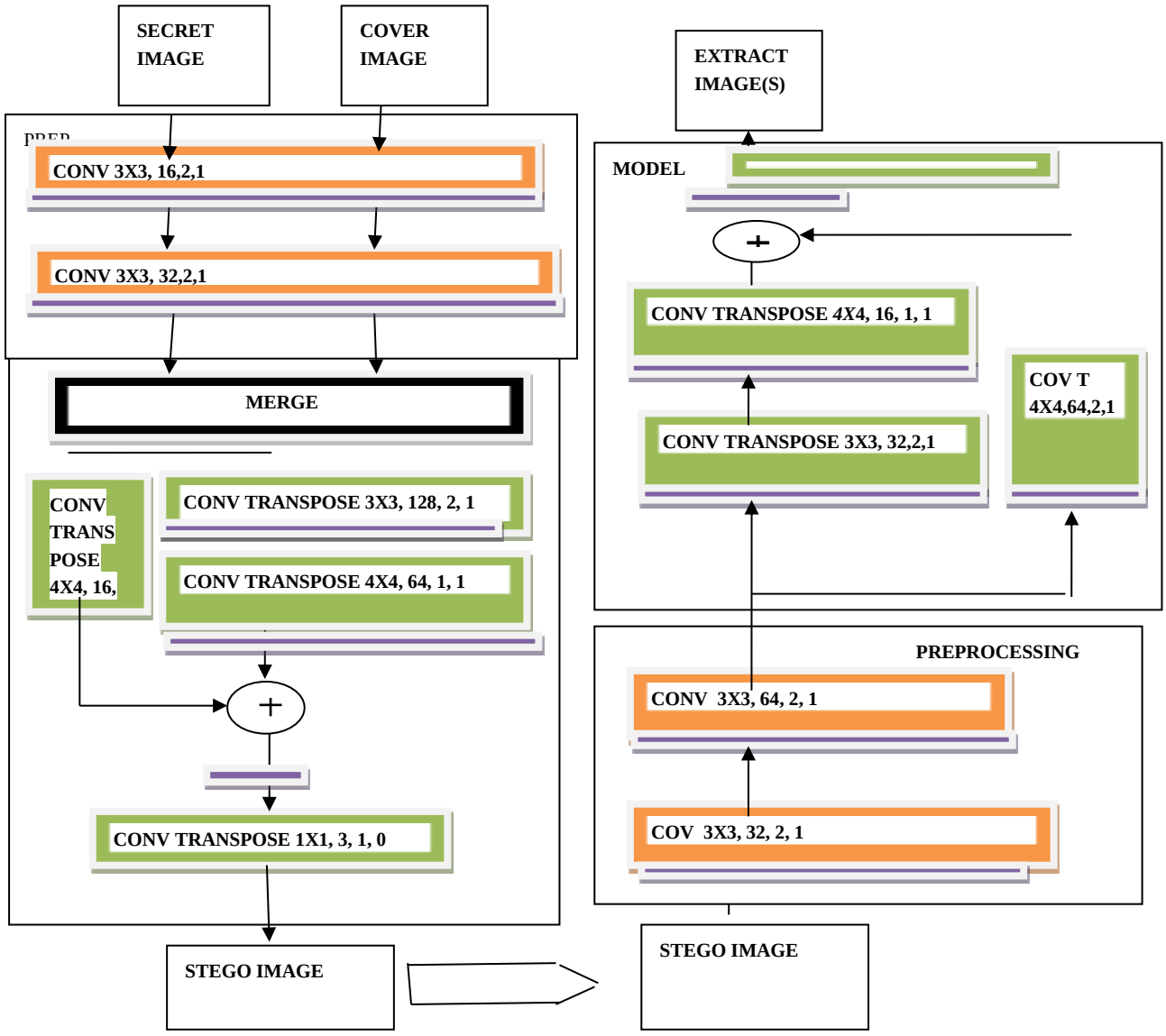

Fig.3. Diagram Architecture of the proposed system

3.2 System Architecture Features

Pre-processing model Convolutional Layers: Extract lower-level local features from the secret, cover, and stego images. These features form the basis for further processing. ResNet Connections: Ensure that meaningful features are maintained between the embedding and extraction phases

enhancing the overall system's performance. Activation Functions (ReLU): Introduce non-linearity, aiding in learning the details of the secret image, cover, and stego images. Batch Normalization (BN): Speed up network training by normalizing input values, improving convergence. Kernel Sizes, Stride, and Padding: Control the size and quality of features extracted by convolutional layers

Operational Model: Convolutional Layers: Generate feature maps and reconstruct images from them during both the embedding and extraction phases. ResNet Connections: Maintain consistency between the embedding and extraction processes, enhancing stego image imperceptibility and extracted image quality. Activation Functions (ReLU): Facilitate feature learning during the transformation of feature maps into images. Batch Normalization (BN): Improve training efficiency during image reconstruction. Kernel Sizes, Stride, and Padding: Control feature size and quality during image reconstruction.

3.3 Design Approach and Evaluation Method

During the design of the system, the following assumption was considered: Combining the use of rsnet50 and autoencoders and intelligent algorithms can improve the image quality and security of the stego image. ResNet50 can extract more robust and reliable features from cover images, thereby improving the accuracy and security [10].

All setup and design were done using python programming language on the Jupyter IDLE. The python language was chosen because of its sufficient supply of open-source libraries and frameworks and modules such as TensorFlow, keras and Scikit-learn, that makes it easy to develop and implement complex machine learning algorithm. The various modules of python which were utilized for the design are listed and expatiated are in section 3.6**.** The design was segmented, albeit recursive, with the improvement done on the preprocessing stage of the system.

a) Peak Signal to Noise Ratio (PSNR): Peak Signal to Noise Ratio (PSNR) is a widely used evaluation metric for image quality assessment. It measures the difference between the original image and the steganographic image. The higher the PSNR value, the better the quality of the steganographic image.

a) Structural Similarity Index (SSIM): The Structural Similarity Index (SSIM) is another evaluation metric for image quality assessment. It measures the structural similarity between the original image and the steganographic image. The higher the SSIM value, the better the quality of the steganographic image.

b) Mean Squared Error (MSE) MSE is a widely used metric to evaluate the quality of steganographic images. It measures the difference between the original and steganographic images. A lower MSE score indicates better image quality. The MSE calculation can be represented as follows:

c) Visual Turing Test (VTT) The VTT is a subjective evaluation method that involves asking humans to distinguish between the original and steganographicimages. This method evaluates the quality of the steganographic images from a human perspective (Wu *et al.*, 2019).

d) The combination of these metrics, it were deployed to evaluate different aspects of the hiding process, including visual similarity, fidelity, and distortion, thereby enabling the development and comparison of more effective steganographic algorithms and techniques.

3.4 Usability Test

Usability test was conducted on the new system to determine the functionality of various modules of the system, in order to ascertain its compliance with the user's expectation. This was done by initializing the new system and testing each module with values and instructions. During this process the behavior of each module was observed and recorded to ensure that every module functioned as expected. Table 4.1 in chapter four shows the tests carried out on the developed system to ascertain its usability and to juxtapose its result with the primary objective earlier stated in chapter one of this report.

Imperceptibility

In general, the stego image must be similar to the cover image and the extracted image must be similar to the secret image. The results of visual effects of the proposed method are measured by the quality of the imperceptibility using PSNR and SSIM. In the case of image steganography, the secret image plays the role of destructive noise, which reduces the quality of the stego image. The steganography algorithm is said to have a good performance, if the PSNR value between cover and stego image are greater than 35 dB and SSIM value must be close to 1. So, higher PSNR and SSIM values between cover and stego image implies smaller distortion after steganography. PSNR results from the calculation of the logarithm of MSE of two images. The formula for PSNR is as follows:

$$PSNR(i,i^{'}) = 10 \times Log_{10}(\frac{Max^2}{MSE(i,i^{'})}) \quad (4)$$

where Max represents the maximum pixel value of the image. Forasmuch as RGB images have 8 bits per pixel, the Max value is 255. SSIM is another measure of the similarity of two images that uses human perception and structural features of images. This criterion is calculated by three principal coefficients: contrast, luminance, and structure. The formula is as follows:

$$SSIM = I(i,i^{'}) \times c(i,i^{'}) \times s(i,i^{'}) \quad (5)$$

$$I(i,i^{'}) = \frac{2\mu_i\mu_{i^{'}} + c_1}{\mu^2{}_i + \mu^2{}_{i^{'}} + c_1} \quad (6)$$

$$c(i,i^{'}) = \frac{2\sigma_i\sigma_{i^{'}} + c_2}{\sigma^2{}_i + \sigma^2{}_{i^{'}} + c_2} \quad (7)$$

$$s(i,i^{'}) = \frac{\sigma_{ii^{'}} + c_3}{\sigma_i\sigma_{i^{'}} + c_3} \quad (8)$$

The first coefficient is (i, i'), a function that compares luminance between two images. The second coefficient compares the contrast of two images, which is notated by (i, i'). The last coefficient is (i, i') that compares structures of two images. Also, μ_i and σ_i are mean and standard deviation pixels of an image, respectively and $\sigma_{ii'}$ is a covariance between image i and image i'. The parameters C_1, C_2, and C_3 are constant values to avoid the zero denominators.

According to the previous studies, the values $C_1 = (0.01 \times 255)^2$, $C_2 = (0.03 \times 255)^2$, and $C_3 = C_2/2$ are recommended as the default values [25]. The range of SSIM is (−1,1), when two images were exactly the same, SSIM takes the value of 1.

Loss function

The higher the visual quality of the stego-image, the less suspicious it becomes, which can increase security. Also, the extracted image must be similar to the secret image. So, we selected mean square error (MSE) to measure the similarity between the two images. MSE is calculated as:

$$MSE(i,i^{'}) = \frac{1}{M \times N \times 0} \sum_{c=1}^{0} \sum_{n=1}^{n} \sum_{m=1}^{m} (i_{m,n,c} - i^{'}_{m,n,c})^2 \quad (9)$$

Where $i_{m,c}$ is the pixel **(m, n)** of image i in channel c. Finally, the overall loss is calculated by:

$$Loss = \alpha \times MSE(c,h) + (1-\alpha) \times MSE(s,e) \quad (10)$$

with the hyper-parameter α, the designer can trade off the visual quality of stego image against the extracted images. If the higher α is selected, the higher the visual quality of the stego image is achieved. In contrast, if the lower α is selected, the higher quality of the extracted image is restored. Because of the differentiable functions, the system can be trained end-to-end.

3.5 The proposed algorithm is as follows:

The schema below further simplifies the algorithm. It represents to the nearest explanation and in simple natural language, the overview of the various processes involved at each step of the developed algorithm.

Step1: Fetch

#Get the cover image from a data source, such as a file or an input stream

Step2: Preprocessing

#Convert Image Format: If needed, convert the cover image to a standardized format suitable for processing, like JPEG or PNG.

#Resize Image: Adjust the dimensions of the image to a desired size suitable for CNN processing.

#Normalize Image: Normalize the pixel values of the image to a common range (e.g., [0, 1]).

Step3: Encoding:

#Convert Secret Data: Convert the secret data into a suitable format for embedding, such as binary or numerical representation.

#Encode Data using CNN: Utilize a CNN-based steganography algorithm to embed the secret data into the cover image.

#Generate Stego-Image: Obtain the stego-image, which contains the embedded secret data.

Step4: Decoding:

#Extract Data using CNN: Apply the CNN-based steganography algorithm to extract the hidden secret data from the stego-image.

#Decode Data: Convert the extracted data back to its original format, if necessary.

Step5: Postprocessing:

#Resize/Normalize Extracted Data: Adjust the size or normalization of the extracted data, if required.

#Data Interpretation: Interpret the extracted data based on its format and intended usage.

Step6: Output:

Display Stego-Image: Show the stego-image with the embedded secret data, if visual representation is needed.

Display/Use Extracted Data: Present or utilize the extracted secret data for further processing or analysis.

Summary of the Algorithm

1. Collect and preprocess data:
    - Load images from a dataset
    - Convert images to binary format
    - Split the dataset into training, validation, and testing sets

- Normalize and resize the images

2. Define the model architecture:
    - Define the CNN encoder architecture
    - Define the autoencoder architecture
    - Define the RSNets decoder architecture
    - Define the loss function and optimizer
3. Train the model:
    - Train the CNN encoder using the training set
    - Train the autoencoder using the encoded images and the original images as targets
    - Train the RSNets decoder using the decoded images as targets
    - Evaluate the model on the validation set and adjust the hyperparameters
4. Test the model:
    - Test the model on the testing set
    - Evaluate the performance of the model using metrics such as accuracy and PSNR
5. Implement the model:
    - Integrate the different models
    - Create a user interface to interact with the model
6. Deploy the system:

Fig. 4. Shows the flowchart of the proposed system

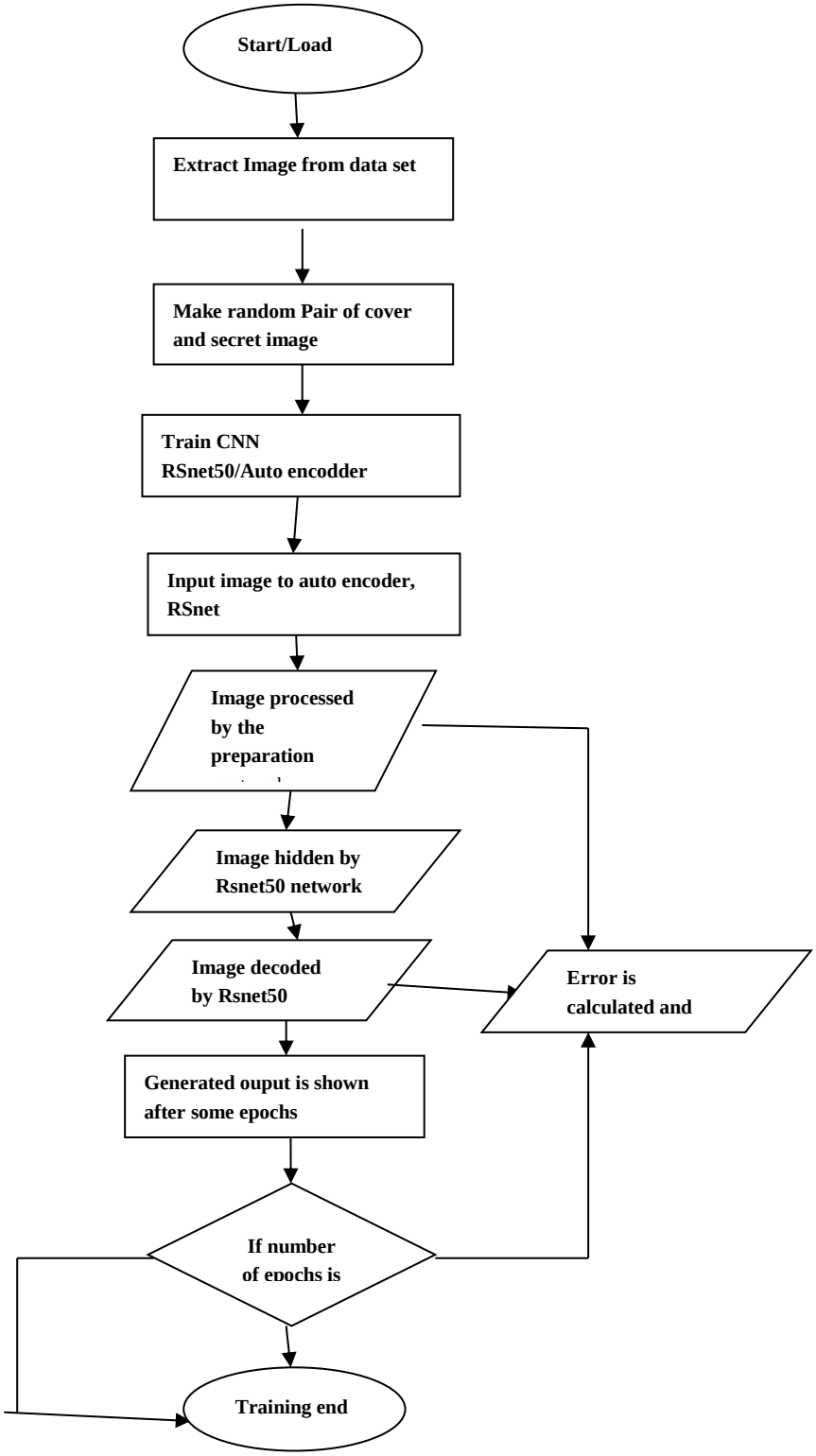

Fig. 4: System Flow Chart

Model Summary

The model summary provides details on the layers, output shapes, and parameters of the steganography model. Here is an excerpt from the model summary:

Model: “Model”

Table1: Summary of the layers

Layer(type)	Output Shape	Param #	Connected to
input_carrier type(Input Layer)	None[(32, 32, 3)]	0	
input_payload type(Input Layer)	None[(32, 32, 3)]	0	
Conv2D_6	None[(32, 32, 16)]	448	input_carrier[0][0]
Conv2D_17	None[(32, 32, 3)]	300	Concatenate_4[0][0]

Total Param = 26,243

Trainable Params = 26, 243

Non- trainable Params = 0

The model consists of multiple convolutional layers and produces three main outputs: the encoded steganographic image and two decoded outputs (secret information and recovered cover image).

Loss Function and Optimization

Loss Function

Two loss functions are defined for training the model:

1. `branched loss function`: This loss function considers the difference between the payload (secret information) and the decoded payload, the difference between the host (cover image) and the encoder output, and the difference between the host and the decoder host output. It combines these differences with customizable weights (alpha, beta, gamma) to calculate the overall loss.

2. `Loss function: This simplified loss function considers only the differences between the payload and the decoded payload and between the host and the encoder output.

Optimization

The model is trained using the Adam optimizer with a learning rate of 0.0001. This optimizer adjusts the model's weights during training to minimize the defined loss function

3.6 Training Process

The model is trained for a total of 250 epochs. Each epoch involves the following steps:

1. Data shuffling: The training and test datasets are shuffled to ensure randomization in each epoch.

2. Normalization: Data normalization is performed to ensure consistency in pixel values between cover images and secret information.

3. Training and Testing: The model is trained using the training dataset, and its performance is evaluated on the test dataset. Training metrics such as loss, PSNR (Peak Signal-to-Noise Ratio), and SSIM (Structural Similarity Index) are computed.

4. Metrics Collection: Metrics, including loss, PSNR, and SSIM, are collected for both training and testing phases and stored for analysis.

Model Weights Saving.

The model weights are a critical part of the steganography model as they capture the learned patterns during training. Saving the model weights is essential for future use, including inference and model retraining. The model weights are saved to the fileweights.h5using the following code:*steganography_model.save_weights('./model_weights/weights.h5')*

This code snippet ensures that the model's weights are persisted for future use.

Training Results Storage

Monitoring the training process is crucial for assessing model performance and identifying potential improvements. Two DataFrames, train_df and test_df, are created to store training and testing metrics, respectively. These metrics include loss, PSNR (Peak Signal-to-Noise Ratio), and SSIM (Structural Similarity Index) for both the host (cover image) and payload (secret information).

train_df = pd.DataFrame(train_metrics, columns=['Epoch', 'Train Loss', 'TrainH PSNR', 'TrainP PSNR', 'TrainH SSIM', 'TrainP SSIM'])

test_df = pd.DataFrame(test_metrics, columns=['Epoch', 'Test Loss', 'TestH PSNR', 'TestP PSNR', 'TestH SSIM', 'TestP SSIM'])

The training and testing metrics are organized into DataFrames and are later saved to an Excel file for further analysis. The Excel file, named training_results.xlsx, is created using the pd.ExcelWriter context manager.

Testing Process with Normalization Modification

Data Preparation for Testing

For testing purposes, random cover images and payload images from the test dataset are selected. These images are normalized using a modified normalization function test_normalize. This normalization function ensures that pixel values are within a specific range, facilitating consistent testing.

Testing Metrics

The steganography model is evaluated using the test dataset, and various metrics are computed, including:

- Test Loss: A measure of the overall loss on the test dataset.
- PSNR-H (Peak Signal-to-Noise Ratio for Host): A metric indicating the quality of the recovered cover images.
- PSNR-P (Peak Signal-to-Noise Ratio for Payload): A metric indicating the quality of the extracted secret information**.**

4. Result and Discussion

4.1 Interfaces and Modules Home Screen

This screen/page displays immediately the system is initialized by a user and the home screen provides an overview of the application and serves as the starting point for users.

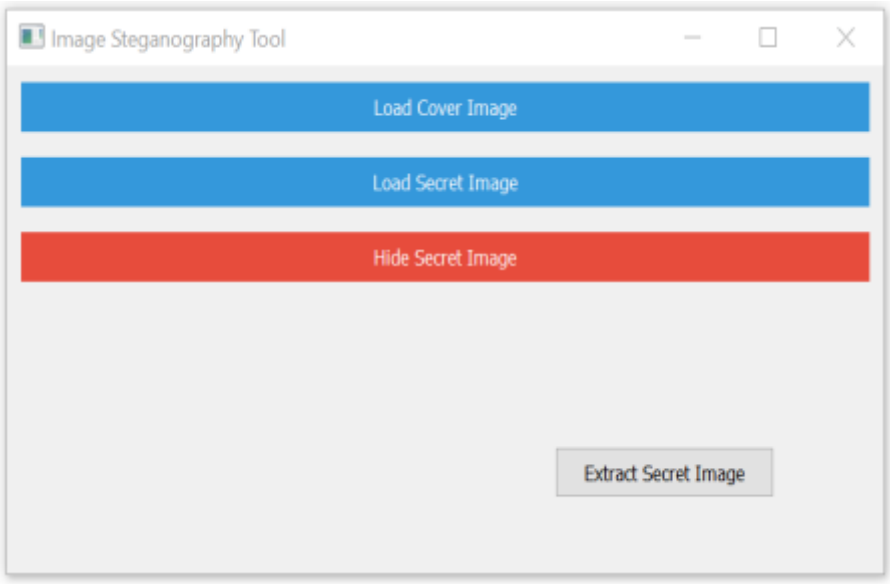

Fig. 5: Software interface (home screen)

1. Load Cover Image Button:

Function: This button allows the user to load or import a cover image from their device. The cover image is the visible or public image in which the secret information will be hidden. Usage: Clicking on this button opens a file explorer or dialog box, enabling the user to select the cover image they want to use for hiding the secret information.

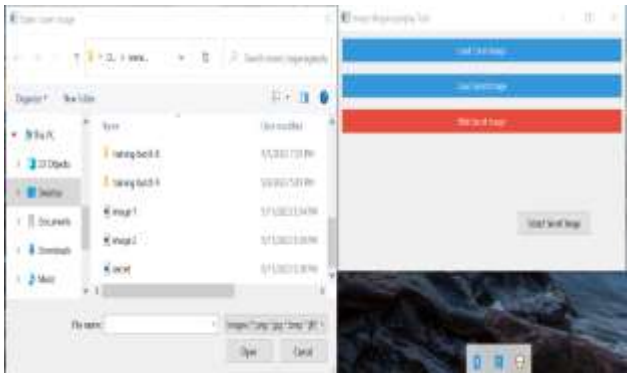

Fig. 6: Uploading cover image

Fig. 7: Cover image uploaded.

2. Load Secret Image Button Function:

This button enables the user to load or select a secret image that they want to hide within the cover image. Usage: Clicking on this button opens a file explorer or dialog box, allowing the user to choose the secret image they wish to conceal within the cover image.

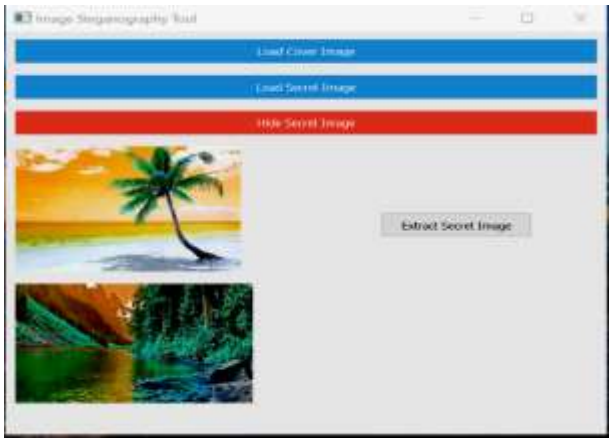

Fig. 8a Uploading secret image

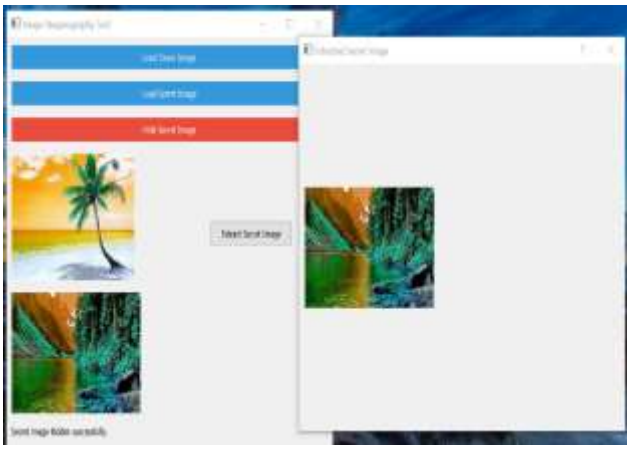

Fig. 8b: Secret image extracted from cover image successfully.

- Function:After loading both the cover image and the secret image, this button initiates the steganography process. It embeds the secret image within the cover image to create a stego image, which appears as the cover image but contains hidden data.
- Usage: Clicking on this button triggers the steganography algorithm to hide the secret image within the selected cover image.

2. Extract Secret Image Button:

- Function: This button is used to extract the hidden secret image from a stego image. It's the reverse process of hiding the secret image and is used when the user wants to retrieve the concealed data.
- Usage: Clicking on this button starts the extraction process on a stego image, revealing the concealed secret image

The software interface home screen provides a user-friendly way to interact with the image steganography system. Users can load both the cover and secret images, hide the secret image within the cover image, and later extract the secret image when needed. These buttons facilitate the entire process of hiding and retrieving concealed information within images

```
from tensorflow.keras.models import Sequential
from tensorflow.keras import Model
from tensorflow.keras.layers import Dense, Flatten, Conv2D, Concatenate, Input

class EncoderNetwork:
    def __init__(self, carrier_shape=(32, 32, 3), payload_shape=(32, 32, 1)):

#         super(EncoderModel, self).__init__()
        self.carrier_shape = carrier_shape
        self.payload_shape = payload_shape

    def _init_branch_payload(self, payload):

        self.branch_payload_conv_1 = Conv2D(16, 3, padding='same', activation='relu', kernel_initializer=
        self.branch_payload_conv_2 = Conv2D(16, 3, padding='same', activation='relu', kernel_initializer=
        self.branch_payload_conv_3 = Conv2D(16, 3, padding='same', activation='relu', kernel_initializer=
        self.branch_payload_conv_4 = Conv2D(16, 3, padding='same', activation='relu', kernel_initializer=
        self.branch_payload_conv_5 = Conv2D(16, 3, padding='same', activation='relu', kernel_initializer=
        self.branch_payload_conv_6 = Conv2D(16, 3, padding='same', activation='relu', kernel_initializer=

        self.payload_tensors = [self.branch_payload_conv_1, self.branch_payload_conv_2,
                                self.branch_payload_conv_3, self.branch_payload_conv_4,
                                self.branch_payload_conv_5, self.branch_payload_conv_6]

    def _init_branch_carrier(self, carrier):

        self.branch_carrier_conv_1 = Conv2D(16, 3, padding='same', activation='relu', kernel_initializer='
        self.branch_carrier_concat_1 = Concatenate()([self.branch_carrier_conv_1, self.branch_payload_con
```

Fig. 9a. (snippet from the encode module)

```
from tensorflow.keras.layers import Concatenate, Conv2D
from tensorflow.keras import Model

class DecoderNetwork:
    def __init__(self, target_image_shape=(32, 32, 1)):
        self.target_image_shape = target_image_shape

    def _init_branch(self, input_, num_filters):
        conv_1 = Conv2D(num_filters, 3, padding='same', activation='relu', kernel_initializer='he_normal')
        concat_1 = Concatenate()([input_, conv_1])
        conv_2 = Conv2D(self.target_image_shape[-1], 3, padding='same', activation='relu', kernel_initiali

        return conv_2

    def get_network(self, encoder_output):
        decoded_output = self._init_branch(encoder_output, num_filters=16)
        decoded_host_output = self._init_branch(encoder_output, num_filters=8)

        return decoded_output, decoded_host_output
```

Fig. 9b. (snippet from the decoder module)

4.2 Results of Usability Test

Table 4.1 describes the various activities done during testing of each module, the processes involved as well as the various outputs and results.

Table 2: Results of Usability Test

SN	• Activity	• Process	• Output/Result
	• Module: Encode	•	•
1	• User uploads cover image	• User selects a cover image, uploads a secret image and initiates the encoding	• Stegonagraphic image (I_stego)
	• Module: Decode	•	•
2	• User uploads a steganographic image	• User selects a steganographic image and initiates a decoding process	• Extracted image

3	• Module: Steganalysis • User uploads steganographic images for analysis	• User selects a steganographic image and initiates steganalysis	• Detection rate, analysis details

Training vs. Test Loss

To evaluate the model's training progress, we created a plot that illustrates the training loss and test loss per epoch. This graph provides insights into the convergence of the model during training. Here are the key observations:

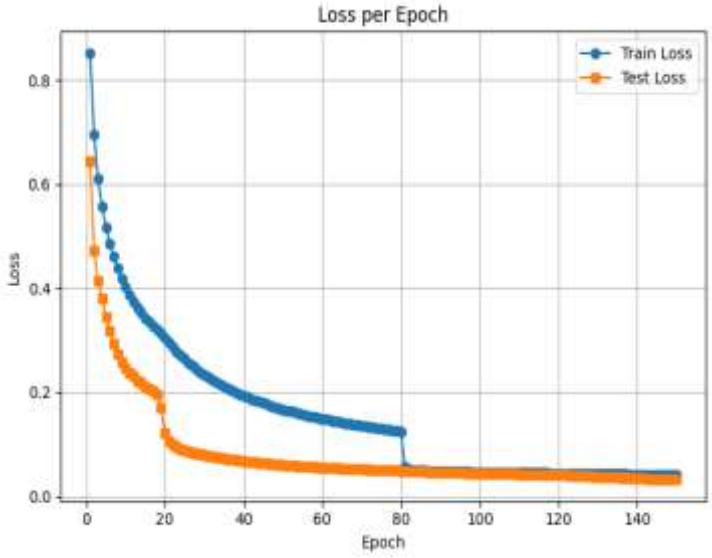

Fig.. 10. Graph for Training vs. Test Loss

- Both training loss and test loss decrease over epochs, indicating that the model is learning and improving its performance.
- The gap between training and test loss remains relatively consistent, suggesting that the model generalizes well to unseen data.

4.3 PSNR and SSIM Metrics

The PSNR and SSIM metrics are essential for assessing the quality of the images generated or processed by the model. These metrics help evaluate image distortion and quality. Here are some findings:

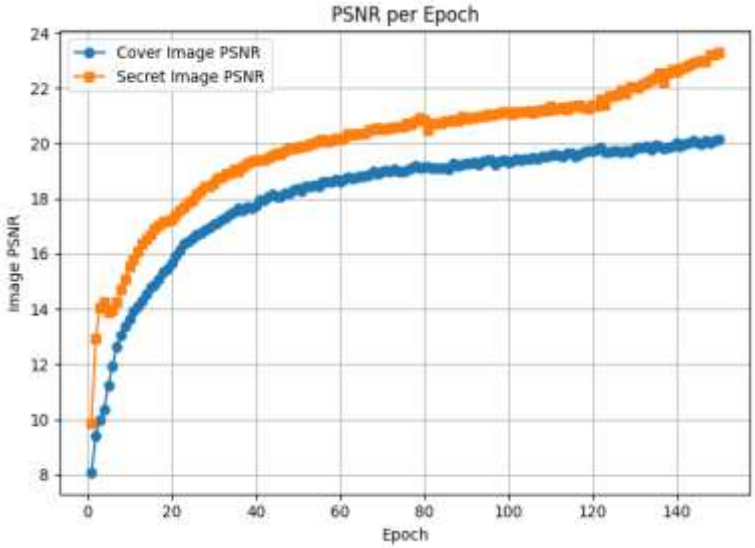

Fig. 11 Graph for PSNR and SSIM metrics

- PSNR values for both cover and secret images generally increase over epochs, indicating a reduction in image distortion.
- SSIM values also improve over epochs, suggesting better image quality and structural similarity.

4.4 Cover SSIM vs. Secret SSIM per Epoch

To evaluate the SSIM values for cover images and secret images, we created a plot that illustrates the SSIM per epoch. This graph allows us to observe how well the model preserves the structural similarity of both cover and secret images during training. Here are the key findings:

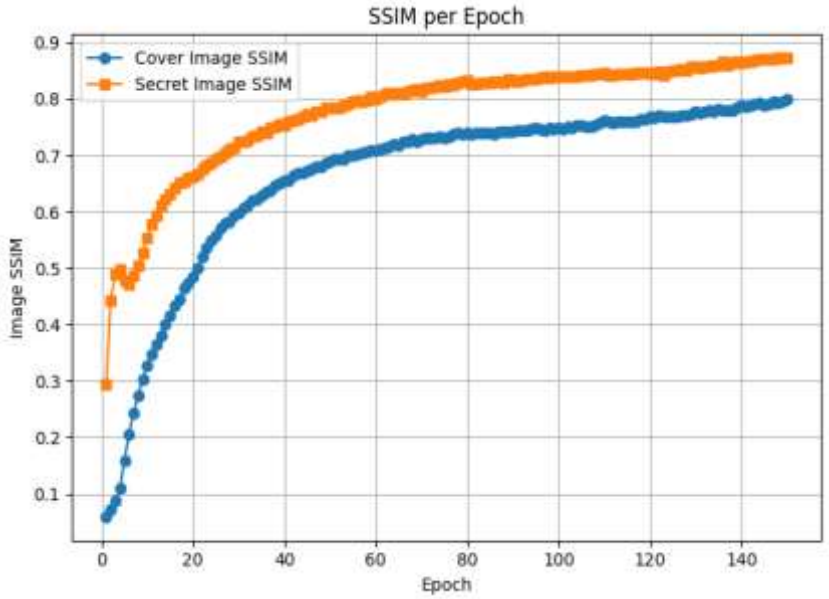

Fig. 12:Graph for Cover SSIM vs. Secret SSIM per Epoch

Cover Image SSIM: The SSIM values for cover images show an upward trend over epochs. This indicates that the model progressively maintains the structural similarity of the cover images, resulting in reduced distortion.

- Secret Image SSIM: Similarly, the SSIM values for secret images exhibit improvement throughout the training process. This suggests that the model effectively conceals the secret information within the images while preserving their structural in Table 2 shows the listed improvements which were achieved by the new system over the existing system.

Table 2 Performance comparison of proposed system with existing system

SN	EXISTING SYSTEM	NEW SYSTEM
1	The existing system relies solely on CNN (CAE) for image steganography, which limits its capability to handle complex image data efficiently.	The new system introduces the power of CNN (CAE) and ResNet in tandem, resulting in a significantly enhanced image steganography algorithm. This combination allows for more effective embedding and extraction of hidden information within images.
2	The existing system may be resource-intensive and less efficient due to its reliance on a single CNN architecture.	The new image steganography system is designed to be more lightweight, ensuring efficient operation even on less powerful devices. This lightweight design enhances accessibility and usability.
3	The existing system might depend on external resources or services, making it less versatile and potentially requiring an internet connection to function.	The new system is fully standalone, eliminating the need for external resources or internet connectivity. This independence enhances its reliability and ensures usability in various environments, including offline scenarios.
4	The existing system may struggle to handle complex image data effectively, limiting its suitability for tasks involving intricate images.	The new system, with the integration of ResNet, demonstrates remarkable capabilities in handling complex image data. It excels in concealing and extracting hidden information within images of varying complexities, making it a robust choice for image steganography tasks.

Conclusion

This study introduces a novel color image steganography approach that leverages the power of deep learning through a fusion of convolutional autoencoders and the ResNet architecture. Traditional steganography techniques have long grappled with critical shortcomings, including limited data capacity, security vulnerabilities, and fragility. In recent years, significant progress has been made by employing convolutional neural networks, particularly autoencoders, to address these challenges in image hiding and extraction. In the proposed method, all images undergo a preprocessing step

using a lightweight convolutional deep neural network, designed for efficient feature extraction. Subsequently, the operational model is employed to create stego images and recover hidden data. This operational model is an autoencoder based on the ResNet structure, which excels at generating images from extracted feature maps. Notably, our approach ensures consistency between the embedding and extraction phases, enhancing the system's reliability. To assess the performance of our method, we conducted experiments using two prominent datasets: Cipher10. We quantitatively compared our results with those of previous related works, evaluating metrics such as Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and hiding capacity. The experimental outcomes demonstrate the superiority of our proposed scheme, highlighting its potential for practical applications on desktop systems due to its lightweight computational demands

References

Sana V. Parveen K, Renjith V. Ravi, Basma Abd El-Rahiem, Mangesh M. Ghonge (2021): Multidisciplinary Approach to Modern Digital SteganographyISBN13: 9781799871606|ISBN10: 1799871606|EISBN13: 9781799871620 DOI: 10.4018/978-1-7998-7160-6.ch002

Fridrich, J., Goljan, M., & Du, R. (2001). Reliable detection of LSB steganography in color and grayscale images. In Proceedings of the 5th Workshop on Information Hiding (pp. 226-244).

Yang, Y., Xu, C., & Shi, Y. Q. (2019). High-Capacity Image Steganography Based on Joint Spatial-Transform Domain Deep Neural Network. IEEE Transactions on Image Processing, 28(3), 1170-1184.

Zuo, S., Zhang, W., Liu, J., & Wang, J. (2020). Optimized steganography method based on deep convolutional autoencoder. IET Image Processing, 14(9), 1806-1813.

Xu, Y., Pan, X., Zhang, Y., Liu, X., Zhang, C., & Wang, Y. (2018). A novel image steganography scheme based on convolutional neural network. Multimedia Tools and Applications, 77(16), 21055-21069.

Wu, H., Xu, X., & Dong, R. (2019). Deep residual network for image steganography. Multimedia Tools and Applications, 78(4), 4393-4407.

Zhang, C., Zheng, X., & Wang, S. (2021). Steganography for images based on residual network and K-Means clustering. Journal of Ambient Intelligence and Humanized Computing, 12(6), 6145-6154.

Sasi, S., & Arunmozhi, M. (2020). Image steganography using convolutional neural networks with quantized representations. Journal of Ambient Intelligence and Humanized Computing, 11(1), 513-526

Wei, J., Wu, S., & Zhang, Y. (2020). Improved image steganography using CNN. In 2020 International Conference on Electronics, Communications and Information Technology (ECIT) (pp. 507-510). IEEE.

Chen, C. (2018). A survey of digital watermarking techniques. Journal of Electronic Imaging, 27(4), 040801