JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Deep steganography using CNN and Machine Learning Techniques

[1]Aayushee Bhatt, [2]Khushi Patel, [3]Jalpa Shah

[1] Lecturer, Department of Computer Engineering, Silver Oak University, Ahmedabad, India
[2] Lecturer, Department of Computer Engineering, Silver Oak University, Ahmedabad, India
[3] Assistant Professor, Department of Computer Engineering, Silver Oak University, Ahmedabad, India

Abstract: — From the list of possible project ideas, steganography was chosen mostly due to the novelty and interest in the subject. One more thing is that it makes sure your data is safe. The security of our information is so well-designed that no one, not even a hacker or intruder who gains access to our system, will be able to access our data, information, or image files. The art of steganography involves hiding data, such as images, audio files, or text, inside another file type. A fundamental idea underpins image-based steganography. The contents of a picture typically the colors of each pixel are defined by the digital data (pixels) that comprise the image. Because we know that there are three values red, green, and blue—for every pixel in a picture. The plan is for the cover picture to completely engulf the one we wish to hide, rendering it invisible to anybody looking in from outside. Creating a sophisticated deep steganography model that can embed one picture (the secret image) into another (the cover image) with little detectability is the main objective of this project. At its core, this goal is to optimize the balance between embedding capacity and visual quality. By incorporating machine learning algorithms, which dynamically adjust the payload based on cover image characteristics, this research not only tackles current information security challenges but also paves the way for a more resilient and adaptable method of protecting sensitive data in the digital age.

Keywords — Deep Learning, Image Processing, Image Hiding, CNN, RNN, Loss-Function, Steganalysis

CHAPTER 1: INTRODUCTION

The theoretical underpinning of this research endeavor, which studies deep steganography employing machine learning approaches for picture concealing, is built in numerous important ideas and academic fields [8]. The word "steganography" comes from the Greek words "steganos" meaning "concealed" and "graphie" meaning "pertaining to writing [8]." It refers to the method of hiding information within other data in order to keep it hidden. Text and pictures are among the data kinds that have traditionally been subject to it, with the goal of achieving clandestine communication or protecting data [8]. Deep learning is a subfield of machine learning that focuses on understanding and representing complex data patterns through the use of artificial neural networks, specifically deep neural networks [3]. A key part of deep learning, convolutional neural networks (CNNs) have shown remarkable performance in picture-related tasks including picture detection and creation. The antithesis of steganography, steganalysis seeks to unearth hidden facts inside data by utilizing a variety of statistical and machine learning methods to spot modifications made by steganographic procedures. Encryption, access control, and steganography are just a few of the many tactics and technologies that fall under the umbrella of information security, which aims to protect data from prying eyes [7]. A revolutionary idea called dynamic payload adjustment uses machine learning algorithms to change the amount of hidden data depending on the data's properties, making it more

secure and harder to detect [11]. In light of the ever-changing field of steganalysis methodologies and computer capabilities, this study is driven by the need to improve the security of steganography. Deep learning gives a chance to construct more robust and flexible steganography models capable of efficiently hiding data inside digital pictures [3]. This research aims to push the frontiers of steganography by combining these theoretical underpinnings, creating a mutually beneficial link between deep learning and data concealment, and then using that to build a state-of-the-art steganography model that is more secure, flexible, and practical.

1.1 PROBLEM STATEMENTS

Limitations of Traditional Steganography Methods Traditional steganography methods exhibit shortcomings concerning security and payload capacity. These limitations undermine their effectiveness in concealing information within digital content. Consequently, there is a pressing need to surpass these constraints and explore innovative approaches to enhance the security and payload capacity of steganographic techniques [1].

Demand for Advanced Steganography Techniques
The increasing sophistication of security threats necessitates the development of advanced steganography techniques. Conventional methods may not be sufficiently robust to withstand evolving detection technologies and adversarial attacks [1]. Addressing this demand requires the exploration and implementation of cutting-edge methodologies that leverage the capabilities of deep learning.

Objective to Develop a Deep Learning-Based Steganography Model
The primary goal is to pioneer a deep learning-based steganography model that excels in concealing one image within another effectively [14]. By harnessing the power of deep neural networks, this research aspires to overcome the limitations of traditional approaches and achieve superior concealment capabilities, ensuring the secure and efficient embedding of information.

Challenges in Embedding Algorithm Optimization and Robustness
This research acknowledges and aims to tackle the challenges associated with optimizing the embedding algorithm for minimal distortion. Striking a balance between effective concealment and preserving the integrity of the cover image poses a significant challenge. Additionally, there is a need to enhance the model's robustness against various attacks, ensuring the hidden information remains secure under diverse scenarios.

Development of a User-Friendly Practical Tool
Recognizing the practical applications of steganography, the research aims to bridge the gap between theoretical advancements and real-world usability. This involves the creation of a user-friendly software tool that facilitates the seamless hiding and extraction of information within images. The tool's user interface will be designed for accessibility, making it applicable to a broad audience, including non-experts in steganography.

1.2 OBJECTIVE

2. Design and Implement a Deep Neural Network Architecture for Steganography

2. Develop a sophisticated deep neural network architecture specifically tailored for steganography purposes. The design should focus on efficiently embedding a secret image into a cover image, utilizing the power and flexibility of deep learning to achieve optimal concealment.

3. Investigate and Optimize the Embedding Algorithm

4. Conduct a comprehensive investigation into the embedding algorithm to ensure it not only hides the secret image effectively but also optimizes the process. Strive to minimize noticeable distortions in the cover image, emphasizing the need for efficiency in the embedding process.

5. Assess Robustness Against Image Processing Operations and Steganalysis Techniques

6. Rigorously test the developed steganography method by subjecting it to common image processing operations like resizing and compression. Additionally, assess its resilience against various steganalysis techniques, ensuring the hidden information remains intact and undetectable under different conditions.

7. Evaluate Security Against Attacks

8. Conduct thorough evaluations to assess the security of the hidden information against potential attacks aimed at revealing the concealed image. Identify and address vulnerabilities to ensure the robustness of the steganography model in preserving the confidentiality of the embedded data.

9. Investigate Techniques to Evade Steganalysis Methods

10. Explore and develop techniques to enhance the model's ability to evade steganalysis methods, making the presence of the secret image more challenging to detect. This involves implementing countermeasures to mitigate potential weaknesses and vulnerabilities.

11. Compare with Existing Methods in Terms of Embedding Capacity, Visual Quality, and Security

12. Conduct a detailed comparative analysis between the proposed deep steganography approach and existing methods. Evaluate and compare embedding capacity, visual quality of the stego-images, and the overall security of the concealed information, providing insights into the superiority of the developed model.

13. Develop a User-Friendly Software Tool

14. Create a user-friendly software tool that leverages the developed steganography model. This tool should enable users to seamlessly hide and extract secret images within cover images. Prioritize user experience and accessibility, ensuring the tool is intuitive and effective for both novice and advanced users.

CHAPTER 2: LITERATURE REVIEW

The literature review on image steganography presents a comprehensive exploration of various techniques and methodologies employed in concealing information within images, each contributing to the evolving landscape of covert communication. [10] introduces the concept of Deep Multi-Image Steganography with Private Keys, emphasizing the use of a private key to conceal multiple images within a single cover image. The effectiveness is demonstrated through jumbled textures and sounds, ensuring authorized access to concealed pictures. [9] proposes Image Steganography by Deep CNN Auto-Encoder Networks, utilizing CNN layers to learn image feature hierarchies and conceal information within the cover image's attributes, showcasing versatility across multiple sources. provides a strategic perspective on steganography and steganalysis, invoking game theory concepts like Nash equilibrium to highlight the competitive dynamics in covert communication strategies.[5]

overview delves into bitmap graphics, emphasizing the integration of hidden data in LSB, leading to a higher proportion of near-duplicate colors[17]. introduces SteganoGAN, a Generative Adversarial Network model, advocating for deep learning's potential in image steganography [15]. presents "Hiding Images in Plain Sight: Deep Steganography," showcasing high-performance steganalysis networks trained as binary classifiers [18]. explores the significance of global information in steganalysis, leveraging global statistical information for improved feature expression [16]. introduces a novel steganography technique using a magic square matrix and affine cipher, achieving high accuracy on the Lena dataset[4]

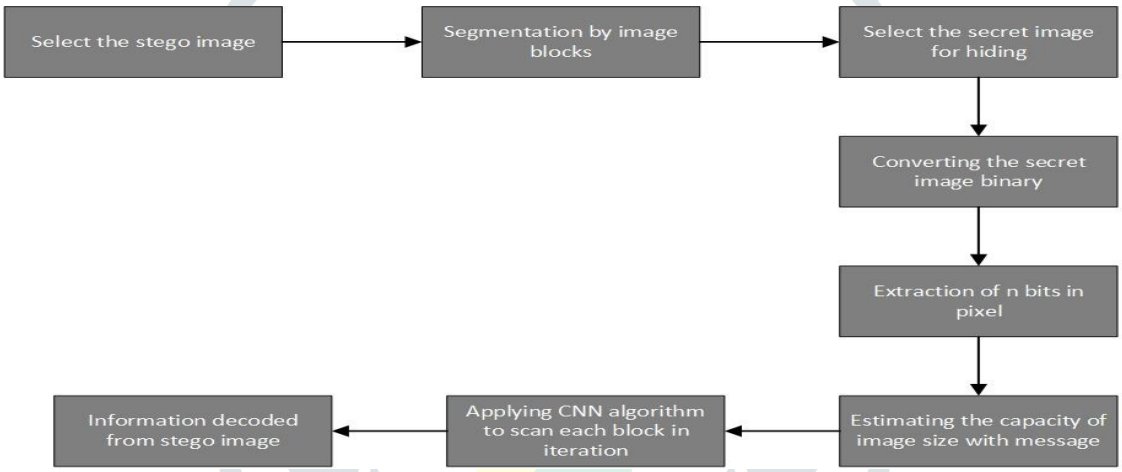
Machine learning's role in steganography detection is highlighted by [6], where the random forest algorithm achieved an accuracy of 82.4%. [13] employs machine learning-based algorithms, including k-nearest neighbors and logistic regression, achieving 79.6% accuracy. [12] introduces an Ensemble Transfer

Learning Model for detecting stego images, achieving an impressive 84% accuracy on CNN. integrates block chain and deep learning in steganography, emphasizing the collision-resistant nature of hash values[19].

Lastly, introduces a High Capacity Image Steganography Method Combined with Image Elliptic Curve Cryptography and Deep Neural Network, employing max-pooling layers for spatial invariance [2]. The study evaluates image quality metrics, showcasing high SSIM and PSNR values. Collectively, these studies reflect the diverse approaches and advancements in image steganography, ranging from deep learning models and machine learning techniques to strategic game-theoretic considerations and novel cryptographic integration.

CHAPTER 3: METHODOLOGY

By methodically creating, optimizing, and assessing a deep steganography technology that is combined with machine learning for the purpose of concealing one picture within another image, the methodology for this research project is constructed to address the aim and objectives of the project.



Data Collection and Preparation: Acquire a diverse dataset comprising both cover images and secret images. Select cover images that represent a wide range of image types and content, ensuring diversity in the dataset. Prepare secret images containing information to be concealed within the cover images. Apply pre-processing techniques, including resizing and normalization, to ensure consistency and facilitate effective training.

Model Selection: Opt for Convolutional Neural Networks (CNNs) due to their proven effectiveness in handling image-related tasks. Consider the inclusion of Recurrent Neural Networks (RNNs) for their ability to handle sequential data, addressing specific requirements in steganography applications.

Training Process: Initiate an iterative training process, systematically adjusting model parameters to optimize the embedding capacity of the deep steganography model. Focus on minimizing distortion in cover images while ensuring the effectiveness of information concealment. Design a loss function tailored to strike a balance between distortion minimization and efficient concealment, ensuring the model's ability to generate stego-images with high visual quality and minimal perceptual differences from the original cover images.

Embedding Algorithm Optimization:
Investigate and refine the embedding algorithm to enhance its efficiency and minimize distortions in the cover images. Implement techniques such as regularization and fine-tuning to achieve an optimal balance between effective concealment and maintaining the integrity of the cover images.

Robustness Assessment:
Subject the developed deep steganography model to a battery of tests, including attacks such as resizing, compression, and steganalysis techniques. Assess the robustness of the model against these challenges, ensuring the concealed information remains secure and undetectable under various conditions.

User-Friendly Software Tool Development:
Develop a user-friendly software tool based on the optimized deep steganography model. Prioritize an intuitive user interface for seamless hiding and extraction of secret images within cover images. Conduct usability testing to ensure the practical applicability of the tool for users with varying levels of expertise in steganography.

Performance Evaluation and Comparison:
Evaluate the performance of the proposed deep steganography approach through quantitative and qualitative analysis. Compare the model's embedding capacity, visual quality, and security metrics with existing methods to showcase its advancements and superiority in the field.

Documentation and Reporting:
Document the entire methodology, including experimental setups, parameter choices, and results. Prepare detailed reports summarizing the findings and contributions made by the developed deep steganography technique integrated with machine learning.

CHAPTER 4: IMPLEMENATATION

Dataset

In the captivating realm of Tiny ImageNet, a mosaic of 100,000 enchanting images unfolds, each a vibrant masterpiece in the palette of 64x64 pixels. Across this visual tapestry, 200 distinct classes emerge, each commanding its own unique identity. Embarking on the journey of training, 500 images per class serve as the artistic foundation, meticulously honing the skills of our digital apprentices. The proving ground of validation beckons with 50 images per class, a crucible for refining the artistry to perfection. Finally, the grand spectacle of testing unveils itself with another set of 50 images per class, an evaluative symphony that resonates with the essence of visual mastery. Step into this diminutive universe where pixels become storytellers, and classes are chapters in the epic saga of Tiny ImageNet – a testament to the boundless creativity captured within the confines of 64x64 brilliance.

Data Pre-Processing

Handling Files: The code starts by ide ntifying the files in the Tiny ImageNet dataset's training and testing folders and importing the libraries that are needed. A preliminary step in data preparation is to create two numpy arrays, x_train and x_test, to hold the picture data. x_train and x_test both contain two thousand photos; the former is used for training and the latter for testing purposes.

The next loop inserts the training dataset (tiny-imagenet-200/train) into x_train with ten photos per class chosen at random. By include pictures from several categories, this method guarantees that the training data is diverse. Pixel value normalisation occurs after the training and test pictures have been loaded by the algorithm. In machine learning, normalisation is an essential pre-processing step for reducing the range of input characteristics to a consistent value. The pixel values are transformed into the interval [0,1] by dividing them by 255.0.

Data Division

The initial 1000 photos make up input_S, while the second 1000 images make up input_C, making a total of two parts to the normalized training data (input_S). It is probable that this separation is for steganographic purposes; for example, input_S may stand for the "secret" data that has to be concealed, while input_C might be the cover material that is utilised to hide it.

To guarantee compatibility with following operations, the data types of input_C and input_S are changed to float64. Making ensuring the data type is consistent across the code is what this phase is all about. When it comes to deep steganography, this code section is essential for pre-processing and preparing picture data. To maximize efficiency during model training and assessment, this procedure entails importing photos, splitting them into training and testing sets, adjusting pixel values to make them more consistent, and organizing the data for steganography.

Message Loss Calculation:

Two loss functions are defined here: rev_loss and full_loss.

The rev_loss function calculates the loss between the true and predicted values. It is a custom loss function that computes the loss as the sum of squared differences between the true and predicted values, scaled by a factor beta. This loss function is useful for optimizing the model's ability to reverse the steganography process, i.e., to extract the hidden message from the container.

The full_loss function calculates the overall loss for the steganography model. It decomposes the input into the message and container components, both for the true and predicted values. The message component consists of the first three channels of the input tensor, while the container component consists of the last three channels.

For the Message Loss Calculation, the rev_loss function is applied to the message components of the true and predicted tensors. This calculates the loss specifically related to the message extraction process.

The Container Loss Calculation is computed as the sum of squared differences between the true and predicted container components. This loss quantifies how well the model preserves the container while hiding the message.

The Overall Loss Calculation is computed as the sum of the message loss and the container loss. This formulation ensures that the model is trained to minimize both the distortion of the message during hiding and the alteration of the container.

In summary, these loss functions are crucial for training the deep steganography model. They enable the model to learn how to effectively hide messages within containers while minimizing distortion and preserving the integrity of the container. The beta parameter in rev_loss allows for adjusting the importance of message extraction during optimization, providing flexibility in training the model based on specific project requirements.

Image Steganography: Hide_Network

The **prep_and_hide_network** function presents a meticulously crafted neural network architecture designed explicitly for image steganography, focusing on the art of concealing a message within an image, known as the "Hiding Image."

Inputs

- **input_message**: This serves as a placeholder for the message intended for concealment.
- **input_cover**: This acts as a placeholder for the cover image, the canvas onto which the message will be discreetly embedded.

Layers

Convolutional Layers: Several convolutional layers are strategically employed to process the input message (input_message). Each of these layers employs a distinct set of filters, with varied sizes (3x3, 4x4, 5x5), capturing features at different spatial scales. Operating in parallel, sets of convolutional layers (x1, x2, x3) skilfully extract diverse aspects from the input message.

Concatenation: The outputs derived from the parallel convolutional layers are harmoniously concatenated (x) to amalgamate the features extracted by filters of different sizes.

Intermediate Processing: To elevate the model's capability in discerning higher-level representations and patterns, additional sets of convolutional layers are thoughtfully applied to the concatenated features (x).

Output Layer: The concluding layer (image_container) comprises a convolutional layer adorned with three filters. This pivotal layer engenders the steganographic image, a fusion of the cover image and the concealed message.

Model Definition

Leveraging the Keras **Model** class, the encoder model is meticulously crafted. It takes **input_message** and **input_cover** as inputs and yields **image_container** as the crowning output. This model is a manifestation of the encoding process in steganography, encapsulating the intricate artistry where the input message surreptitiously intertwines with the cover image, giving rise to the steganographic masterpiece.

Return: The function gracefully returns the encoder model, encapsulating the neural network architecture meticulously tailored for the subtle art of preparing and concealing messages within cover images.
In essence, this function unveils a neural network model meticulously honed for the realm of image steganography. Through adept utilization of convolutional layers, it seamlessly processes input messages, skillfully intertwining them with cover images to birth steganographic compositions, thus becoming an instrumental tool for concealing messages within the visual fabric of images.

Image Steganography: Reveal Network

The reveal_network function introduces a meticulously crafted neural network architecture designed to unveil hidden messages from steganographic images. Let's dissect its structure and functionalities:

Inputs

- **reveal_input**: This serves as a placeholder for the steganographic image, the canvas from which the concealed message is to be extracted.

Noise Injection: The introduction of GaussianNoise(0.01) stands as a pivotal layer, infusing Gaussian noise with a standard deviation of 0.01 into the steganographic image. This strategic step enhances the model's robustness to variations and noise in the input image, potentially elevating its generalization capabilities.

Layers

Convolutional Layers: Much like the encoding network, this architecture employs multiple convolutional layers to process the input steganographic image (reveal_input). These layers diligently extract features crucial for the revelation of hidden messages. Operating in parallel, distinct sets of convolutional layers (x1, x2, x3) utilize various filter sizes (3x3), capturing features at diverse spatial scales.

Concatenation: The outputs derived from parallel convolutional layers are harmoniously concatenated (x) to amalgamate the features extracted by filters of different sizes.
Intermediate Processing: To discern higher-level representations and patterns within the steganographic image, additional sets of convolutional layers are thoughtfully applied to the concatenated features (x).
Output Layer: The conclusive layer (message) encompasses a convolutional layer adorned with three filters. This pinnacle layer brings forth the revealed message, extracted from the steganographic image.

Model Definition

Leveraging the Keras **Model** class, the reveal network is meticulously defined. It takes **reveal_input** as input and yields **message** as the crowning output. This model mirrors the intricate decoding process in steganography, where the concealed message is skillfully extracted from the steganographic image.

Fixed Noise (Optional): The **fixed** parameter, a Boolean flag, determines whether the injected noise remains constant during training. If set to True, the injected noise stays fixed, potentially contributing to regularization and stability.

In summary, this function meticulously crafts a neural network model tailored specifically for the revelation of hidden messages from steganographic images. Leveraging convolutional layers, it adeptly processes steganographic images, extracting concealed messages and embodying the essence of the decoding process in steganography. The injection of Gaussian noise and the strategic use of parallel convolutional layers contribute to the model's precision and robustness in accurately revealing messages from steganographic canvases.

Image Steganography: Model Compilation

The code segment establishes a comprehensive deep steganography model by seamlessly incorporating the previously defined encoder (prep_and_hide_network) and decoder (reveal_network) networks into a unified architecture, tailored for both training and inference. Let's delve into the intricacies of each step:

Input Definitions:

- **shape = input_S.shape[1:]:** Extracts the shape of the input data (input_S), delineating the dimensions of the images engaged in the steganography process.
- **input_message and input_container**: These placeholders, with shapes determined by shape, signify the inputs for message and container images.

Network Instantiation:

- **prep_and_hide = prep_and_hide_network(shape):** Instantiates the encoder network (prep_and_hide_network) with the specified input shape.
- **reveal = reveal_network(shape):** Similarly, instantiates the decoder network (reveal_network) with the same input shape.

Compilation

- **reveal.compile(optimizer='adam', loss=rev_loss):** Compiles the decoder network (reveal) utilizing the Adam optimizer and the rev_loss function. This configuration readies the decoder for training by defining the optimization algorithm and the loss function to minimize.

Freezing the Decoder

- **reveal.trainable = False**: Freezes the weights of the decoder network (reveal). By setting trainable to False, the decoder's weights remain unaltered during training, preserving its fixed state within the entire deep steganography model.

Model Composition

- **output_container = prep_and_hide([input_message, input_container]):** Computes the output container image by routing the message and container images through the encoder network (prep_and_hide).
- **output_message = reveal(output_container):** Determines the output message by passing the output container image through the decoder network (reveal).

Model Definition

- **deep_stegan=Model(inputs=[input_message,input_container], outputs=concatenate([output_message, output_container])):**
  Constructs the deep steganography model, employing the Keras Model class. This definition specifies the inputs (message and container images) and outputs (revealed message and container

images). The outputs, derived from the decoder (output_message) and encoder (output_container), are concatenated along the channel axis, culminating in the final output.

Compilation of the Deep Steganography Model:

- **deep_stegan.compile(optimizer='adam', loss=full_loss):** Finally, compiles the deep steganography model (deep_stegan) with the Adam optimizer and the full_loss function. This setup equips the model for training, specifying the optimization algorithm and loss function to minimize during the training process.

In summary, this code segment meticulously orchestrates the establishment of the deep steganography model, seamlessly integrating encoder and decoder networks into a cohesive architecture primed for both training and inference. The model's compilation with appropriate optimizers and loss functions ensures its readiness for the intricate training journey.

Model Summary

```
Model: "model_2"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 64, 64, 3)]       0         []

 input_2 (InputLayer)        [(None, 64, 64, 3)]       0         []

 model (Functional)          (None, 64, 64, 3)         293273    ['input_1[0][0]',
                                                                  'input_2[0][0]']

 model_1 (Functional)        (None, 64, 64, 3)         155938    ['model[0][0]']

 concatenate_13 (Concatenat  (None, 64, 64, 6)         0         ['model_1[0][0]',
 e)                                                               'model[0][0]']

==================================================================================================
Total params: 449211 (1.71 MB)
Trainable params: 293273 (1.12 MB)
Non-trainable params: 155938 (609.13 KB)
```

Figure No. 4.1 Model Summary

The code segment orchestrates the training loop for the deep steganography model, seamlessly integrating a dynamic learning rate scheduler (lr_schedule) to adapt the learning rate during the training process. Let's delve into the intricacies of each step:

Learning Rate Schedule Function (lr_schedule):

- This function takes an epoch index as input and dynamically determines the learning rate based on predefined schedules.
- It commences with a higher learning rate of 0.001 for the initial 200 epochs (epoch_idx < 200).
- Following 200 epochs, the learning rate is gracefully reduced to 0.0003 for the subsequent 200 epochs (epoch_idx < 400).
- Similarly, it further decreases the learning rate to 0.0001 for the subsequent 200 epochs (epoch_idx < 600).
  Beyond 600 epochs, a lower learning rate of 0.00003 is set for the remaining epochs.

Initialization

- The variable m is initialized with the number of samples in the training set (input_S).
- An empty list, loss_history, is initialized to capture and store the loss values during training.
- batch_size is set to 32, determining the size of each training batch.
- Training Loop (for epoch in range(1000):):

- The loop gracefully unfolds over 1000 epochs to train the model.

Within each epoch:

- Training data (input_S and input_C) undergo shuffling to introduce randomness.
- The number of iterations (itera) is computed based on the chosen batch size.
- Losses (f_loss and r_loss) are accumulated and averaged (f_loss_mean and r_loss_mean) over each batch.

For each batch:

- Message and cover images are carefully selected for the batch.
- The container image is derived by passing the batch through the encoder (prep_and_hide.predict).
- The complete steganography model (deep_stegan) undergoes training on the batch, aiming to minimize the loss between the predicted and actual message and container images.
- The decoder (reveal) undergoes training on the container image to minimize the loss between the revealed message and the actual message.Learning rates for both the steganography model and the decoder are updated using the learning rate schedule (lr_schedule(epoch).

Printing and Logging

- Loss values for each batch and epoch are intelligently printed to monitor the training process.
- Mean losses for each epoch are also displayed for a comprehensive overview.
- Learning rate updates for each epoch are included in the printout.

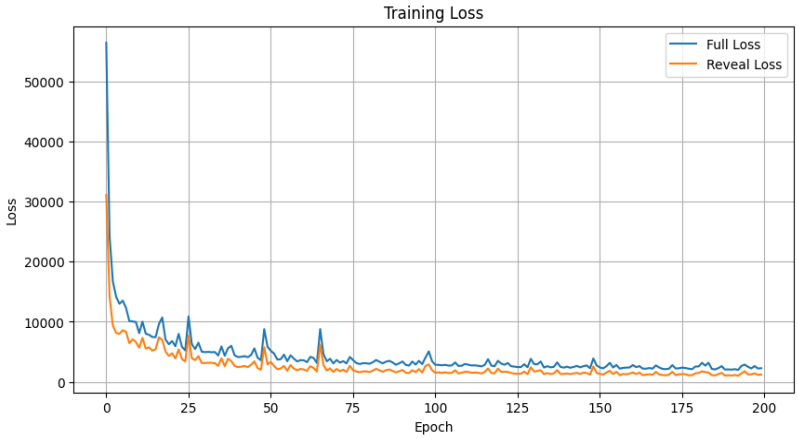


Figure No. 4.2. Model Summary

In summary, this code segment orchestrates a meticulous training loop for the deep steganography model, integrating a dynamic learning rate schedule to optimize the model's performance throughout the training journey. Through iterations of epochs and batches, it intricately updates model parameters to minimize the specified loss function.

EVALUTION OF THE MODEL

Syntax

```
n = 6

def rgb2gray(rgb):
    return np.dot(rgb[...,:3], [0.299, 0.587, 0.114])

def show_image(img, n_rows, n_col, idx, gray=False, first_row=False, title=None):
    ax = plt.subplot(n_rows, n_col, idx)
    if gray:
        plt.imshow(rgb2gray(img), cmap = plt.get_cmap('gray'))
    else:
        plt.imshow(img)
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    if first_row:
        plt.title(title)

plt.figure(figsize=(14, 15))
rand_indx = [random.randint(0, 1000) for x in range(n)]
# for i, idx in enumerate(range(0, n)):
for i, idx in enumerate(rand_indx):
    n_col = 6 if SHOW_DIFF else 4

    show_image(input_C[idx], n, n_col, i * n_col + 1, gray=SHOW_GRAY, first_row=i==0, titl

    show_image(input_S[idx], n, n_col, i * n_col + 2, gray=SHOW_GRAY, first_row=i==0, titl

    show_image(decoded_C[idx], n, n_col, i * n_col + 3, gray=SHOW_GRAY, first_row=i==0, ti

    show_image(decoded_S[idx], n, n_col, i * n_col + 4, gray=SHOW_GRAY, first_row=i==0, ti


    if SHOW_DIFF:
        show_image(np.multiply(diff_C[idx], ENHANCE), n, n_col, i * n_col + 5, gray=SHOW_G

        show_image(np.multiply(diff_S[idx], ENHANCE), n, n_col, i * n_col + 6, gray=SHOW_G

plt.show()
```

This code snippet is designed to visually evaluate the performance of the deep steganography model by displaying a grid of original cover and secret image pairs alongside their corresponding decoded images. The function **show_image** is utilized to display images in a grid layout using Matplotlib. Random indices are selected to showcase a subset of samples for evaluation. For each selected index, the original cover and secret images are displayed, followed by their decoded counterparts. Additionally, if specified, the difference images between the original and decoded images are shown to highlight any discrepancies.

This visualization provides a qualitative assessment of the model's ability to accurately conceal and reveal secret messages within cover images, aiding in understanding the effectiveness and fidelity of the deep steganography approach.

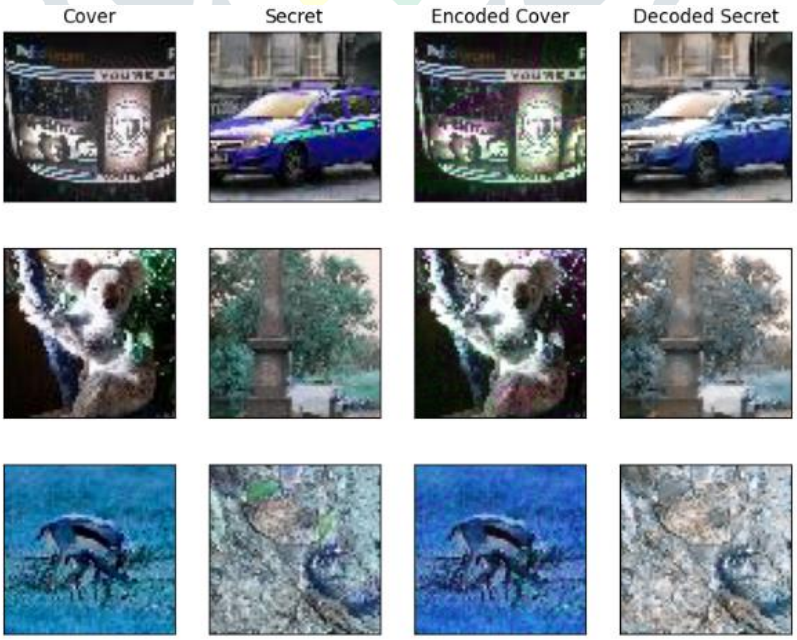


Figure No. 4.3 effectiveness and fidelity of the deep steganography

In conclusion, the deep steganography model outlined in the provided description is a meticulously crafted technology combining machine learning and steganography to conceal one image within another. The methodology encompasses a systematic approach from data collection and preparation to model selection, training, and optimization. Convolutional Neural Networks (CNNs) and Recurrent Neural

Networks (RNNs) are chosen for their effectiveness in handling image-related tasks and sequential data, respectively.

The model undergoes an iterative training process with a focus on minimizing distortion in cover images while ensuring effective information concealment. Custom loss functions, including rev_loss and full_loss, are designed to strike a balance between distortion minimization and efficient concealment. The beta parameter in rev_loss allows for adjusting the importance of message extraction during optimization.

The model consists of an encoder (prep_and_hide_network) tailored for image steganography, effectively concealing messages within cover images, and a decoder (reveal_network) designed to reveal hidden messages from steganographic images. The integration of these networks into a unified architecture for training and inference, along with freezing the decoder's weights, ensures a comprehensive deep steganography model.

The training loop incorporates a dynamic learning rate scheduler, adapting the learning rate during training. The model's evaluation involves quantitative metrics such as PSNR, SSIM, and message recovery accuracy, as well as qualitative assessment through visual inspection. Robustness testing, computational efficiency evaluation and security analyses contribute to a thorough understanding of the model's performance, reliability, and suitability for practical applications.

CHAPTER 5: CONCLUSION

In conclusion, this research endeavors to address critical issues and challenges within the realm of steganography, emphasizing the need for advancements beyond the limitations of traditional methods. The identified problem statements underscore the vulnerabilities associated with conventional steganography techniques, urging the exploration of innovative approaches for enhanced security and payload capacity. The rising demand for advanced steganography techniques, driven by evolving security threats and detection technologies, further accentuates the need for cutting-edge methodologies.

The research's primary objective is to pioneer a deep learning-based steganography model, leveraging the capabilities of deep neural networks to overcome traditional constraints and achieve superior concealment capabilities. The identified challenges in optimizing the embedding algorithm for minimal distortion and ensuring robustness against various attacks highlight the complexity of the task at hand. The goal is not only to effectively hide secret images but also to preserve the integrity of the cover image under diverse scenarios.

A user-friendly practical tool is envisioned as the bridge between theoretical advancements and real-world usability. This tool aims to facilitate the seamless hiding and extraction of information within images, catering to a broad audience, including non-experts in steganography. The defined objectives provide a structured roadmap for achieving these aspirations.

The proposed objectives encompass the design and implementation of a deep neural network architecture tailored for steganography, optimization of the embedding algorithm, robustness testing against image processing operations and steganalysis techniques, evaluation of security against potential attacks, exploration of techniques to evade steganalysis methods, and a comparative analysis with existing methods.

In pursuing these objectives, the research aims to contribute significantly to the field of steganography by introducing a sophisticated and secure deep learning-based model. The envisioned user-friendly software tool will facilitate practical applications of the developed steganography approach, making strides towards bridging the gap between theoretical advancements and real-world utility. Ultimately, this research seeks to advance the state of the art in steganography, providing a foundation for secure and efficient information embedding within digital content.

REFERENCES

[1] Mikołaj Płachta: Detection of Image Steganography Using Deep Learning and Ensemble Classifiers, mdpi, 2022.

[2] Dina Yousif Mikhail: An Ensemble Transfer Learning Model for Detecting StegoImages, mdpi, 2023.

[3] Hyeokjoon kweon, jinsun park: Deep Muliti-image steganography keys, MDPI, 2021.

[4] Kich: Image Steganography by Deep CNN Auto-Encoder Networks, International Journal of Advanced Trends in Computer Science and Engineering, 2020.

[5] Ms. Ayushi Chaudhary: A Novel Approach to Block chain and Deep Learning in the field of Steganography, International Journal of INTELLIGENT SYSTEMS AND APPLICATIONS INENGINEERING, 2023.

[6] Farhan, Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data, 2021.

[7] M. Hassaballah: Deep Learning in steganography and steganalysis from 2015 to 2018, International Journal of Engineering Science and Technology, 2020.

[8] Jammi ashok: International Journal of Engineering Science and Technology,Elsevier Inc, 2019.

[9] Nandhini Subramanian1: Image Steganography: A Review of the Recent Advances, IEEE access, 2019.

[10] Ying Zou: Research on image steganography analysis based on deep learning, Journal of visual communiction and image representation, 2020.

[11] Waleed S. Hasan Al-Hasan:A New Steganography Technique Using Magic Square Matrix and Affine Cipher, Universiti Teknologi Malaysia, 2020.

[12] Hussein Ali Al-Iedane: Applying and Evaluating Machine Learning Models for the Detection of Digital Image Steganography, Basra University for oil and Gas, 2020.

[13] XINTAO DUAN: A New High Capacity Image Steganography Method Combined With Image Elliptic Curve Cryptography and Deep Neural Network, IEEE Access, 2020.

[14] Dhawan, Analysis of various data security techniques of steganography: A survey. Information Security Journal A Global Perspective, 2020.

[15] Kato,A Pre-processing by Using Multiple Steganography for Intentional Image Down sampling on CNN-Based Steganalysis. IEEE, Volume 8, 2020.

[16] Lin, Anew Steganography Method for Dynamic GIF Images Basedon Palette Sort. Hindawi, Volume 10, 2020.

[17] Rao, Image Steganography Analysis Based on Deep Learning. International Information and Engineering Technology Association, 2020.

[18] Subramanian, Image Steganography: A Review of the Recent Advances. IEEE, Volume 8, 2019.