Image Steganography and Steganalysis

**Abstract:** Steganography is the term given to the process of embedding secret data in a different data. In this work, we are presenting a model to embed full-size colour image(s) inside a different image. Convolutional neural networks have been trained together to create the embedding and extracting processes. The proposed model has been trained on the dataset of images randomly drawn from the Tiny-Images database. Other than presenting the practical application of deep learning for steganography and steganalysis, we carefully examine how the result is affected in the cases of single image steganography, double image steganography and triple image steganography. Generally, in steganography, the secret images are embedded within the LSBs of the cover image whereas in this work the secret images have been represented across all the available bits of the other image.

**Keywords:** Steganography, Steganalysis, Deep learning, Convolutional neural network.

# Introduction:

Steganography is the practice of concealing secret or confidential information within non- secret or innocuous carrier media, such as images, audio files, videos, or text documents. It aims to hide the existence of the hidden data, making it difficult for an observer to detect its presence.

The primary objective of steganography is to ensure secure and covert communication, where the hidden information is embedded in such a way that it appears as regular, unmodified data to anyone who is not the intended recipient. It provides an additional layer of security by hiding the existence of the communication itself, making it less susceptible to interception or unauthorized access.

Steganographic techniques involve modifying the carrier media in a subtle manner to hide the secret data. This can be achieved by manipulating the least significant bits (LSBs) of digital files, exploiting the properties of images or audio signals, or using mathematical transformations and encoding schemes. The embedded data is often encrypted to provide an additional layer of confidentiality.

It's important to note that while steganography provides a means of hiding information, it does not ensure the security or integrity of the hidden data. Additional encryption and security measures may be required to protect the confidentiality and authenticity of the embedded information. Moreover, steganography can be subject to detection and analysis by skilled steganalysts, who employ various statistical, visual, and machine learning-based techniques to uncover hidden data.

In summary:

## transport medium= cover message + secret message(s)

Steganography differs from cryptography in the following ways as describes below:

**Table 1: Difference between Steganography and cryptography**

|  |  |
| --- | --- |
| **Steganography** | **Cryptography** |
| It is about hiding the „existence‟ of the  secret message. | It is about hiding the „meaning‟ of the secret  message. |
| Concern in steganography is regarding the embedding capacity and detectability of the  cover image. | Concern in cryptography is regarding the robustness against being able to be  deciphered. |
| Any type of digital media can act as a  carrier or a cover. | Depends upon text as the carrier. |
| Key is optional in steganography. | Key is necessary in cryptography. |

Steganalysis is the process of extracting the presence of steganography in images and attempting to recover the hidden information. It involves analyzing the stego-images to identify any traces left by the steganography.

Steganalysis techniques can be broadly classified into two categories:

1. Statistical Analysis: This approach focuses on analyzing statistical properties of the images to detect deviations caused by steganography. It involves examining features like pixel value distributions, color histograms, or noise characteristics. Statistical analysis aims to identify anomalies that are indicative of the presence of hidden data.
2. Structural Analysis: This approach involves examining the structural properties of the

image to detect steganographic modifications. It includes analyzing file headers, metadata, or specific patterns known to be used in steganography. Structural analysis looks for inconsistencies or unusual patterns that could indicate the presence of hidden information.

In the proposed work, we have developed a convolutional neural network-based model for hiding one, two and three secret images within a cover image and then retrieving the hidden image(s) from the cover image. The dataset used has been picked from the “Tiny images dataset. Our model consists of three sections (Preparation Network, Hiding Network and Reveal Network).

## Single Image Steganography:

The secret image is made to pass through the preparation network, which adjusts the dimensions of image to be hidden match the dimension of the image to be used as cover in case the secret image’ original size is smaller than the dimension of the cover image and transforms the colour-based pixels to more useful features for succinctly encoding the image

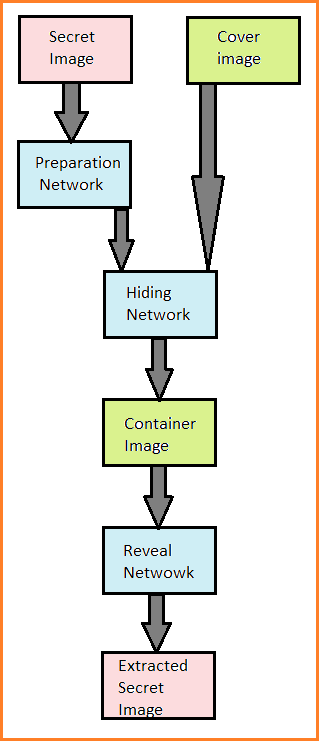
Then the result of the preparation network and image to be covered are together passed through

the hiding network, which works on the result of preparation network(s) and the image to be covered and then combines two to create the carrier image or container image.

The result of the hiding network is called the container image.

The container image is used by the receiver of the container image. The section is used by the receiver to extract the secret data in the form of image from the container image.

**Fig 2: Flowchart for single image steganography.**



## Double Image Steganography:

The two secret images are made to pass through the two preparation networks respectively, which increase their dimensions to match the dimension of the image to be covered in case the secret

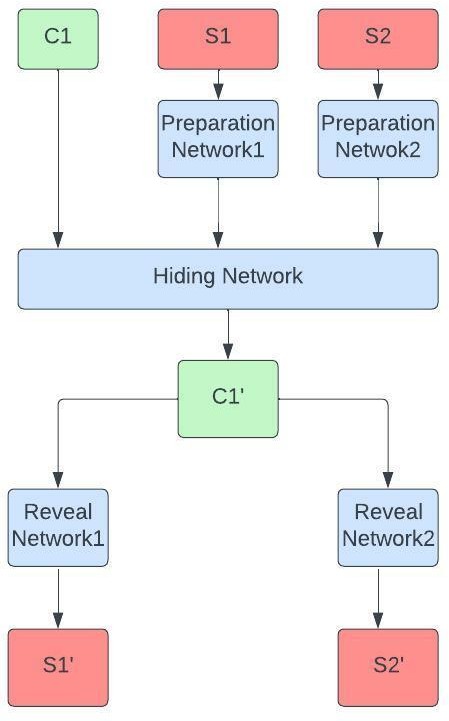
Image’s original size is smaller than the size of the cover image and transforms the colour- based pixels to more useful features for succinctly encoding the image.

Then the outputs of the preparation networks and the cover image are together passed through the hiding network, which takes as input the outputs of the preparation networks and the cover image and then combines the three to create the carrier image or the container image.

The output of the hiding network is called the container image.

The container image is used by the receiver of the container image. The section is used by the receiver to extract the secret data in the form of two secret images from the container image.

**Fig 3: Flowchart for double image steganography.**



**Table 6: Description of symbols used in the diagram**

|  |  |
| --- | --- |
| **C1** | Cover Image |
| **S1** | First Secret image |
| **S2** | Second Secret image |
| **C1’** | Container image |
| **S1’** | First Secret image after retrieval |
| **S2’** | Second Secret image after retrieval |

## Triple Image Steganography:

The three secret images are made to pass through the three preparation networks respectively, which increase their dimensions to match the dimension of the cover image in case the

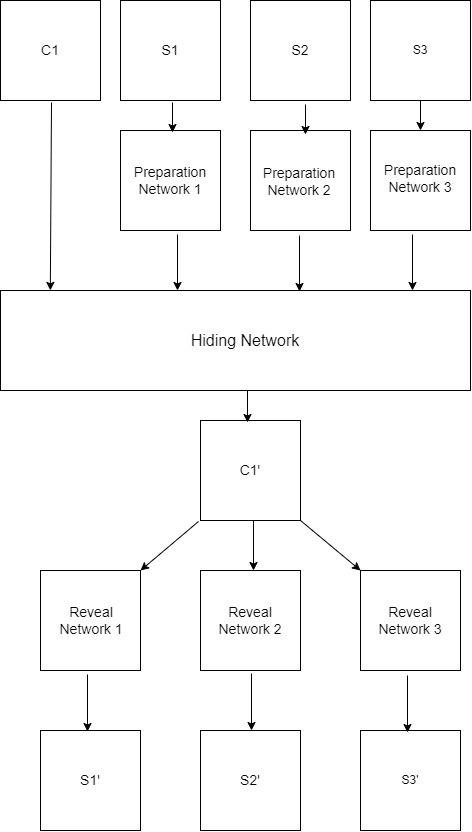
Image’s to be hiddenoriginal size is smaller than the size of the cover image and transforms the colour- based pixels to more useful features for succinctly encoding the image.

Then the outputs of the preparation networks and the image to be covered are together passed through the hiding network, which works on the outputs of the preparation networks and the image to be covered and then combines the four to create the carrier image or the container image.

The output of the hiding network is called the container image.

The container image is used by the receiver of the container image. The section is used by the receiver to extract the secret data in the form of three secret images from the container image.

**Fig 4: Flowchart for triple image steganography.**



**Table 7: Description of symbols used in the diagram**

|  |  |
| --- | --- |
| **C1** | Cover Image |
| **S1** | First Secret Image |
| **S2** | Second Secret Image |
| **S3** | Third Secret Image |
| **C1’** | Container image |
| **S1’** | First secret image after retrieval |
| **S2’** | Second secret image after retrieval |
| **S3’** | Third secret image after retrieval |

There can be some good commercial applications of steganography as mentioned below:

1. Secure Communication: Image steganography can be used for covert communication and secure data exchange. Secret messages can be embedded within images and shared through various communication channels, providing a hidden channel for confidential information.
2. Digital Watermarking: Image steganography can be employed for digital watermarking, where invisible information is embedded within images to verify authenticity, prove ownership, or protect against unauthorized use or copyright infringement.
3. Covert Operations and Intelligence: Image steganography finds applications in covert operations and intelligence activities. Secret messages can be hidden within innocent-looking images to communicate sensitive information or instructions without arousing suspicion.
4. Digital Forensics: Steganalysis plays a crucial role in digital forensics to detect and analyze the presence of hidden information within digital media. It helps investigators identify steganographic techniques used in illegal activities, such as concealing illicit content or communication.
5. Content Filtering and Copyright Protection: Steganalysis techniques can be utilized to identify and filter out images or multimedia files that contain hidden or unauthorized content. This helps protect against the distribution of illegal or copyrighted materials.
6. Cybersecurity and Intrusion Detection: Steganalysis is used in cybersecurity to detect covert communication channels or hidden malware that may be embedded within images. It helps in identifying potential security breaches and intrusion attempts.

# Objective of the proposed work:

1. To develop a deep learning model for maintaining the secrecy and integrity of multiple secret images and protecting them from unauthorised access.
2. To embed the secret images in such a way that their existence itself gets hidden, unlike cryptography which hides only the meaning of the data.
3. To cover the secret images, we use steganography and then retrieve the secret images at the receiver's end by means of steganalysis, in a way that there is minimum loss in reconstruction of the cover image and the hidden images.

# Brief overview of the proposed work:

1. In the proposed work, we have developed a convolutional neural network-based model for hiding two secret images within a cover image and then retrieving the hidden image from the cover image.
2. The dataset used has been picked from the “Tiny images dataset”.
3. Our model consists of three sections (Preparation Network, Hiding Network and Reveal Network).
4. The preparation network adjusts the dimensions of the images which are to be hidden to match the dimension of the image to be covered.
5. The hidden network section works on outputs of the first network and the image to be coveredand then combines the three to create the carrier image or container image.
6. The reveal network is the final section of the model. It is used by the receiver of the container image. The section is used by the receiver to extract the secret data in the form of an image from the container image.

# 2. Literature review:

While remarkable works have been done in the area [2-4], there are still some persisting challenges. One of these challenges is that the hiding process during steganography can disturb the appearance of the transport medium. The amount of disturbance in the transport medium depends upon two factors: The amount of the data to be hidden which is measured in bits per pixel (bpp). The bigger the information, the greater is bpp and the worse the

disturbance in the transport medium [4].

While many significant works have been done by the use of deep neural networks in steganalysis [8-10], the use of deep neural network in the hiding process itself (steganography) still needs more research work [11-14].

**Table 2: Literature Review**

|  |  |  |
| --- | --- | --- |
| **Author/Reference** | **Method Used** | **Application/Work** |
| N. F. Johnson and S. Jajodia [15] | Substitution method | The data is changed to a different form. The container data is looked at to find out the less significant bits in the  distorted area. |
| S. Gupta, G. Gujral, and N. Aggarwal [16] | Substitution method | This method considering that changing some pixel values doesn’t make much change that is  visible |
| R. Das and T. Tuithung [17] | Substitution method | The substitution method is performed very. |
| Wu et al. [18-19] | CNN-based method | The substitution method is  performed very carefully. |
| X. Duan [20] | CNN-based method | GANs which are quite effective were used. |
| A. Singh and H. Singh [22] | GAN-based method | It has been suggested to use GAN to create an end-to-end picture steganographic technique. |

Following are some of the common obstacles faced in training a model for steganography and steganalysis:

1. Finding an appropriate dataset is difficult. No reliable benchmark dataset is provided. The most popular dataset, Tiny-Images, has a significant flaw in the size of the images. The photos are small in size, measuring 64x64.
2. Because the evaluation metrics applied by various approaches vary, it is difficult to compare the suggested method to state-of-the-art methods.

# Material and Methods:

* 1. **Dataset:**

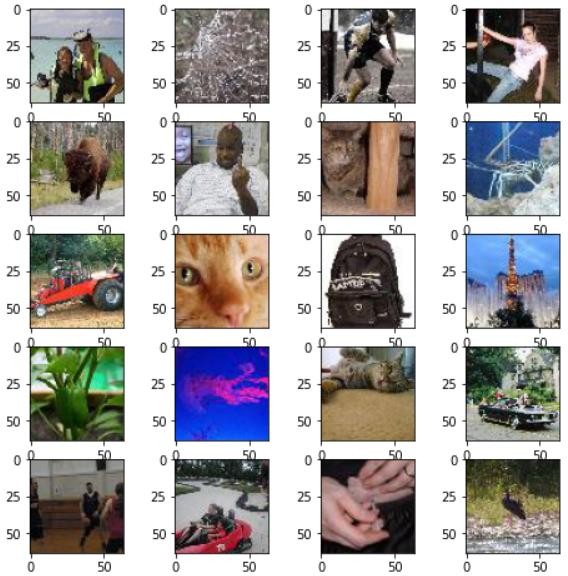
The dataset used has been picked from “Tiny images dataset” [22]. Our dataset consists of 4000 images. Each image is of 64x64x3 dimension. Pairs of secret images and cover images have been made.

For single image steganography, we get 2000 such pairs of secret and cover images after pairing.

For double image steganography, we get about 1322 such pairs of secret and cover images after pairing.

For triple image steganography, we get about 1000 such pairs of secret and cover images after pairing.

**Fig 1: Sample images from the dataset.**



# Proposed Algorithm:

Though parallels are drawn between cryptography and steganography, our work is more similar to image compression done using auto encoders. We aim to compress the secret data from the covert image into the least noticeable areas of the cover image.

Our models for single image steganography, double image steganography and triple image steganography consist of three sections. They are trained together as a single network. Each section’s purpose has been described below individually.

**Step I: Data Collection**

The dataset used has been picked from “Tiny images dataset” [22]. The dataset consists of 4000 images. Each image is of 64x64x3 dimension. Pairs of secret images and cover images have been made.

**Step II: Data Pre-Processing**

1. For single image steganography, 2000 pairs of secret and cover images have been formed.
2. For double image steganography, 1322 pairs of secret and cover images have been formed.
3. For triple image steganography, 1000 pairs of secret and cover images have been formed.

**Step III: Proposed Model**

1. **For Single Image Steganography and steganalysis**: Following processes of “preparation”, “hiding” and “extraction” are to be performed in sequence.
2. **Preparation Network Architecture**:

(1) ‘**conv\_prep0\_3x3’** with **number of filters=50**, **kernel size= (3,3)** and **stride=(1,1)** is applied to the secret image and the resultant is stored in x1.

(2) ‘**conv\_prep0\_4x4’** with **number of filters=10**, **kernel size= (4,4)** and **stride=(1,1)** is applied to the secret image and the resultant is stored in x2.

(3) ‘**conv\_prep0\_5x5’** with **number of filters=5**, **kernel size= (5,5)** and **stride=(1,1)** is applied to the secret image and the resultant is stored in x3.

(4) **x1, x2** and **x3** are concatenated and the resultant image is stored in x.

(5) The same sequence of the arrangement of the layers is repeated one more time on the resultant image with, their results are contatenated with cover image and stored in x.

**Table 3: Layers and Parameters in preparation network:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Layers** | **No. of filters** | **Kernal Size** | **Strides** | **Padding** | **Activation** |
| **conv prep0\_3x3** | 50 | (3, 3) | (1, 1) | same | selu |
| **conv**  **prep0\_4x4** | 10 | (4, 4) | (1, 1) | same | selu |
| **conv prep0\_5x5** | 5 | (5, 5) | (1, 1) | same | selu |

1. **Hiding Network Architecture**:

(1) ‘**conv\_prep0\_3x3’** with **number of filters=50**, **kernel size= (3,3)** and **stride=(1,1)** is applied to x and the resultant is stored in x4.

(2) ‘**conv\_prep0\_4x4’** with **number of filters=10**, **kernel size= (4,4)** and **stride=(1,1)** is applied to x and the resultant is stored in x5.

(3) ‘**conv\_prep0\_5x5’** with **number of filters=5**, **kernel size= (5,5)** and **stride=(1,1)** is applied to x and the resultant is stored in x6.

(4) x4, x5 and x6 are concatenated and the resultant image is stored in x.

(5) The same sequence of the arrangement of the layers is repeated four times on the resultant image with different number of filters in the convolutional layer.

(6) A Conv layer with **3 filters** and **kernel** **size= (3, 3)** is applied on x and the result is stored in **output\_Cprime(carrier image).**

**Table 4: Layers and Parameters in Hiding network:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Layers** | **No. of filters** | **Kernal Size** | **Strides** | **Padding** | **Activation** |
| **conv prep0\_3x3** | 50 | (3, 3) | (1, 1) | same | selu |
| **conv**  **prep0\_4x4** | 10 | (4, 4) | (1, 1) | same | selu |
| **conv prep0\_5x5** | 5 | (5, 5) | (1, 1) | same | selu |

1. **Extraction Network Architecture**:
2. Gaussian Noise is added to reveal image and the result is stored in input\_with\_noise
3. ‘**conv\_prep0\_3x3’** with **number of filters=50**, **kernel size= (3,3)** and **stride=(1,1)** is applied to the carrier image and the resultant is stored in x1.

(2) ‘**conv\_prep0\_4x4’** with **number of filters=10**, **kernel size= (4,4)** and **stride=(1,1)** is applied to the carrier image and the resultant is stored in x2.

(3) ‘**conv\_prep0\_5x5’** with **number of filters=5**, **kernel size= (5,5)** and **stride=(1,1)** is applied to the carrier image and the resultant is stored in x3.

(4) **x1, x2** and **x3** are concatenated and the resultant image is stored in x.

(5) The same sequence of the arrangement of the layers is repeated one more time on the resultant image with, their results are concatenated and stored in x.

(6) A Conv layer with 3 filters and **kernel size= (3, 3)** is applied on x and the result is stored in **output\_Sprime(extracted image).**

**Table 5: Layers and Parameters in Reveal network:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Layers** | **No. of filters** | **Kernal Size** | **Strides** | **Padding** | **Activation** |
| **conv prep0\_3x3** | 50 | (3, 3) | (1, 1) | same | selu |
| **conv prep0\_4x4** | 10 | (4, 4) | (1, 1) | same | selu |
| **conv prep0\_5x5** | 5 | (5, 5) | (1, 1) | same | selu |

1. **For Double Image Steganography and steganalysis**: Following processes of “preparation”, “hiding” and “extraction” are to be performed.

1. **Preparation Network Architecture**:

(1) ‘**conv\_prep0\_3x3’** with **number of filters=50**, **kernel size= (3,3)** and **stride=(1,1)** is applied to the secret image and the resultant is stored in x1.

(2) ‘**conv\_prep0\_4x4’** with **number of filters=10**, **kernel size= (4,4)** and **stride=(1,1)** is applied to the secret image and the resultant is stored in x2.

(3) ‘**conv\_prep0\_5x5’** with **number of filters=5**, **kernel size= (5,5)** and **stride=(1,1)** is applied to the secret image and the resultant is stored in x3.

(4) **x1, x2** and **x3** are concatenated and the resultant image is stored in x.

(5) The same sequence of the arrangement of the layers is repeated one more time on the resultant image with, their results are concatenated with cover image and stored in x.

1. **Hiding Network Architecture**:

(1) ‘**conv\_prep0\_3x3’** with **number of filters=50**, **kernel size= (3,3)** and **stride=(1,1)** is applied to x and the resultant is stored in x4.

(2) ‘**conv\_prep0\_4x4’** with **number of filters=10**, **kernel size= (4,4)** and **stride=(1,1)** is applied to x and the resultant is stored in x5.

(3) ‘**conv\_prep0\_5x5’** with **number of filters=5**, **kernel size= (5,5)** and **stride=(1,1)** is applied to x and the resultant is stored in x6.

(4) x4, x5 and x6 are concatenated and the resultant image is stored in x.

(5) The same sequence of the arrangement of the layers is repeated four times on the resultant image with different number of filters in the convolutional layer.

(6) A Conv layer with **3 filters** and **kernel** **size= (3, 3)** is applied on x and the result is stored in **output\_Cprime(carrier image).**

1. **Extraction Network Architecture**:
2. Gaussian Noise is added to reveal image and the result is stored in input\_with\_noise

(2) ‘**conv\_prep0\_3x3’** with **number of filters=50**, **kernel size= (3,3)** and **stride=(1,1)** is applied to the carrier image and the resultant is stored in x1.

(3) ‘**conv\_prep0\_4x4’** with **number of filters=10**, **kernel size= (4,4)** and **stride=(1,1)** is applied to the carrier image and the resultant is stored in x2.

(4) ‘**conv\_prep0\_5x5’** with **number of filters=5**, **kernel size= (5,5)** and **stride=(1,1)** is applied to the carrier image and the resultant is stored in x3.

(5) **x1, x2** and **x3** are concatenated and the resultant image is stored in x.

(6) The same sequence of the arrangement of the layers is repeated one more time on the resultant image with, their results are contatenated and stored in x.

(7) A Conv layer with 3 filters and **kernel size= (3, 3)** is applied on x and the result is stored in **output\_Sprime(extracted image).**

1. **For Triple Image Steganography and steganalysis**: Following processes of “preparation”, “hiding” and “extraction” are to be performed.

1. **Preparation Network Architecture**:

(1) ‘**conv\_prep0\_3x3’** with **number of filters=50**, **kernel size= (3,3)** and **stride=(1,1)** is applied to the secret image and the resultant is stored in x1.

(2) ‘**conv\_prep0\_4x4’** with **number of filters=10**, **kernel size= (4,4)** and **stride=(1,1)** is applied to the secret image and the resultant is stored in x2.

(3) ‘**conv\_prep0\_5x5’** with **number of filters=5**, **kernel size= (5,5)** and **stride=(1,1)** is applied to the secret image and the resultant is stored in x3.

(4) **x1, x2** and **x3** are concatenated and the resultant image is stored in x.

(5) The same sequence of the arrangement of the layers is repeated one more time on the resultant image with, their results are concatenated with cover image and stored in x.

1. **Hiding Network Architecture**:

(1) ‘**conv\_prep0\_3x3’** with **number of filters=50**, **kernel size= (3,3)** and **stride=(1,1)** is applied to x and the resultant is stored in x4.

(2) ‘**conv\_prep0\_4x4’** with **number of filters=10**, **kernel size= (4,4)** and **stride=(1,1)** is applied to x and the resultant is stored in x5.

(3) ‘**conv\_prep0\_5x5’** with **number of filters=5**, **kernel size= (5,5)** and **stride=(1,1)** is applied to x and the resultant is stored in x6.

(4) x4, x5 and x6 are concatenated and the resultant image is stored in x.

(5) The same sequence of the arrangement of the layers is repeated four times on the resultant image with different number of filters in the convolutional layer.

(6) A Conv layer with **3 filters** and **kernel** **size= (3, 3)** is applied on x and the result is stored in **output\_Cprime(carrier image).**

1. **Extraction Network Architecture**:
2. Gaussian Noise is added to reveal image and the result is stored in input\_with\_noise

(2) ‘**conv\_prep0\_3x3’** with **number of filters=50**, **kernel size= (3,3)** and **stride=(1,1)** is applied to the carrier image and the resultant is stored in x1.

(3) ‘**conv\_prep0\_4x4’** with **number of filters=10**, **kernel size= (4,4)** and **stride=(1,1)** is applied to the carrier image and the resultant is stored in x2.

(4) ‘**conv\_prep0\_5x5’** with **number of filters=5**, **kernel size= (5,5)** and **stride=(1,1)** is applied to the carrier image and the resultant is stored in x3.

(5) **x1, x2** and **x3** are concatenated and the resultant image is stored in x.

(6) The same sequence of the arrangement of the layers is repeated one more time on the resultant image with, their results are contatenated and stored in x.

(7) A Conv layer with 3 filters and **kernel size= (3, 3)** is applied on x and the result is stored in **output\_Sprime(extracted image).**

**Step IV: Training Process:**

Adam Optimizer has been used for the training process with the following hyperparameters:

1. **Number of Epochs-** 250
2. **Learning Rate-** Learning rate has been set tot with 0.001 till first 100 epochs, then decreasing to 0.0003 from 100 epochs to 250 epochs.

# Experimental work:

## 4.1. Implementation Details:

The training details that have been used in each case have been explained below:

1. Adam optimizer has been used for optimization purpose.
2. Learning rate has been set tot with 0.001 till first 100 epochs, then decreasing to 0.0003 from 100 epochs to 250 epochs.
3. Models have been trained for 250 epochs.
4. Tiny Image Dataset has been used, where images are 64x64.
5. The loss used is for the full model is represented as:

## k1||c1 − c1’||^2 + k2||s1 − s1’||^2 + k3||s2-s2'||^2+k4||s3-s3’||

**(C1: Original cover image, C1’: Cover image after steganography, S1: First Original Secret Image, S1’: First Secret Image obtained after steganalysis, S2: Second Original Secret Image, S2’: Second Secret Image obtained after steganalysis, S3: Third Original Secret Image, S3’: Third Secret Image obtained after steganalysis)**

1. Initially we take k1, k2, k3 and k4 as 1.0.

# Results and Discussion:

The performance of the model is evaluated by the value of the loss function.

## Loss Function:

Our loss function is composed of two terms:

Loss in the reconstruction of cover image is **||C − C'||^2,**

and the Loss in the reconstruction loss of secret images is **||S-S'||^2.**

**(C: Original cover image, C’: Cover image after steganography, S: Original Secret Image, S’: Secret Image obtained after steganalysis)**

Loss For Single Images Steganography:

## ||c1 – c1’||^2 + |s1 − s1'||^2

Loss For Double Images Steganography:

## ||c1 − c1’||^2 + ||s1 − s1’||^2 + ||s2-s2'||^2

Loss For Triple Images Steganography:

## ||c1 − c1’||^2 + ||s1 − s1’||^2 + ||s2-s2'||^2+ ||s3-s3'||^2

**(C1: Original cover image, C1’: Cover image after steganography, S1: First Original Secret Image, S1’: First Secret Image obtained after steganalysis, S2: Second Original Secret Image, S2’: Second Secret Image obtained after steganalysis, S3: Third Original Secret Image, S3’: Third Secret Image obtained after steganalysis;)**

**Table 8: Description of the losses for single image steganography**

|  |  |
| --- | --- |
| **1.Loss of the model** | 126547.56 |
| **2. Loss secret1** | 61734.72 |
| **3. Loss Cover** | 64812.84 |

**(Loss secret1= ||Original Secret image – Secret image obtained post steganalysis||^2; Loss Cover= ||Original cover image – Cover image after steganography||^2;**

**Loss of the model= Loss secret1 + Loss Cover)**

## For Double Image Steganography:

**Table 9: Description of the losses for double image steganography**

|  |  |
| --- | --- |
| **1.Loss of the model** | 212927.22 |
| **2. Loss secret1** | 71435.27 |
| **3. Loss secret2** | 62181.72 |
| **4. Loss Cover** | 79310.23 |

**(Loss secret1= ||Original Secret image1 – Secret image1 obtained post steganalysis||^2; Loss secret2= ||Original Secret image2 – Secret image2 obtained post steganalysis||^2; Loss Cover= ||Original cover image – Cover image after steganography||^2;**

**Loss of the model= Loss secret1 + Loss secret2 + Loss Cover)**

## For Triple Image Steganography:

**Table 10: Description of the losses for triple image steganography**

|  |  |
| --- | --- |
| **1. Loss of the model** | 283369.93 |
| **2. Loss secret1** | 73016.59 |
| **3. Loss secret2** | 65051.05 |
| **4.Loss secret3** | 70651.48 |
| **5. Loss Cover** | 74651.29 |

**(Loss secret1= ||Original Secret image1 – Secret image1 obtained post steganalysis||^2; Loss secret2= ||Original Secret image2 – Secret image2 obtained post steganalysis||^2; Loss secret3= ||Original Secret image3 – Secret image3 obtained post steganalysis||^2; Loss Cover= ||Original cover image – Cover image after steganography||^2;**

**Loss of the model= Loss secret1 + Loss secret2 + Loss secret3 + Loss Cover)**

# 5. Conclusion:

Through the proposed work, a method for effective steganography and steganalysis has been presented using deep learinng. The proposed model has further improved the single image steganography shared by (Baluja, 2017) by designing two and three reveal networks for double and triple images steganography and steganalysis.

The performace of the models for the single image, double images and triple images steganography and steganalysis have been evaluated based upon the value of the loss function. For single image steganography, the value of the loss function obtained after training process was 126547.56, for double images steganography, the value of the loss function obtained was 212927.22, and for triple images steganography, it was 283369.93.

The model is successfully being able to embed the secret images at the sender‟s end and being able to successfully retrieve them at the receiver‟s end without much distortion in either

the secret images or the cover image.

## REFERENCES

1. Gary C Kessler. An overview of steganography for the computer forensics examiner. Forensic Science Communications, 6(3), 2014.
2. Jessica Fridrich and Miroslav Goljan. Practical steganalysis of digital images: State of the art. In Electronic Imaging 2002, pages 1–13. International Society for Optics and Photonics, 2002.
3. Jussi Parikka. Hidden in plain sight: The stagnographic image. <https://unthinking.photography/themes/fauxtography/hidden-in-plain-sight-the-steganographic-image>, 2017
4. Hamza Ozer, Ismail Avcibas, Bulent Sankur, and Nasir D Memon. Steganalysis of audio based on audio quality metrics. In Electronic Imaging 2003, pages 55–66. International Society for Optics and Photonics, 2003
5. Farzin Yaghmaee and Mansour Jamzad. Estimating watermarking capacity in gray scale images based on image complexity. EURASIP Journal on Advances in Signal Processing, 2010(1):851920, 2010.
6. Jessica Fridrich, Miroslav Goljan, and Rui Du. Detecting lsb steganography in color, and gray-scale images. IEEE multimedia, 8(4):22–28, 2001.
7. Abdelfatah A Tamimi, Ayman M Abdalla, and Omaima Al-Allaf. Hiding an image inside another image using variable-rate steganography. International Journal of Advanced Computer Science and Applications (IJACSA), 4(10), 2013.
8. Yinlong Qian, Jing Dong, Wei Wang, and Tieniu Tan. Deep learning for steganalysis via convolutional neural networks. In SPIE/IS&T Electronic Imaging, pages 94090J–94090J. International Society for Optics and Photonics, 2015.
9. Lionel Pibre, Jérôme Pasquet, Dino Ienco, and Marc Chaumont. Deep learning is a good steganalysis tool when embedding key is reused for different images, even if there is a cover source mismatch. Electronic Imaging, 2016(8):1–11, 2016.
10. Lionel Pibre, Pasquet Jérôme, Dino Ienco, and Marc Chaumont. Deep learning for steganalysis is better than a rich model with an ensemble classifier, and is natively robust to the cover source-mismatch. arXiv preprint arXiv:1511.04855, 2015.
11. Sabah Husien and Haitham Badi. Artificial neural network for steganography. Neural Computing and Applications, 26(1):111–116, 2015.
12. Imran Khan, Bhupendra Verma, Vijay K Chaudhari, and Ilyas Khan. Neural network based steganography algorithm for still images. In Emerging Trends in Robotics and Communication Technologies (INTERACT), 2010 International Conference on, pages 46–51. IEEE, 2010.
13. V Kavitha and KS Easwarakumar. Neural based steganography. PRICAI 2004: Trends in Artificial Intelligence, pages 429–435, 2004.
14. Alexandre Santos Brandao and David Calhau Jorge. Artificial neural networks applied to image steganography. IEEE Latin America Transactions, 14(3):1361–1366, 2016
15. N. F. Johnson and S. Jajodia, ‘‘Exploring steganography: Seeing the unseen,’’ Computer, vol. 31, no. 2, pp. 26–34, Feb. 1998.
16. S. Gupta, G. Gujral, and N. Aggarwal, ‘‘Enhanced least significant bit algorithm for image steganography,’’ Int. J. Comput. Eng. Manage., vol. 15, no. 4, pp. 40–42, 2012.
17. R. Das and T. Tuithung, ‘‘A novel steganography method for image based on Huffman encoding,’’ in Proc. 3rd Nat. Conf. Emerg. Trends Appl. Comput. Sci., Mar. 2012, pp. 14–18.
18. P. Wu, Y. Yang, and X. Li, ‘‘Image-into-image steganography using deep convolutional network,’’ in Proc. Pacific Rim Conf. Multimedia. Cham, Switzerland: Springer, 2018, pp. 792–802.
19. P. Wu, Y. Yang, and X. Li, ‘‘StegNet: Mega image steganography capacity with deep convolutional network,’’ Future Internet, vol. 10, no. 6, p. 54, Jun. 2018
20. X. Duan, K. Jia, B. Li, D. Guo, E. Zhang, and C. Qin, ‘‘Reversible image steganography scheme based on a U-Net structure,’’ IEEE Access, vol. 7, pp. 9314–9323, 2019.
21. A. Singh and H. Singh, ‘‘An improved LSB based image steganography technique for RGB images,’’ in Proc. IEEE Int. Conf. Electr., Comput. Commun. Technol. (ICECCT), Mar. 2015, pp. 1–4
22. Yuan, C., Wang, H., He, P. *et al.* GAN-based image steganography for enhancing security via adversarial attack and pixel-wise deep fusion. *Multimed Tools Appl* **81**, 6681–6701 (2022). <https://doi.org/10.1007/s11042-021-11778-z>

<https://www.kaggle.com/datasets/aryashah2k/mit-80-million-tiny-images-datase>