AI-powered Adaptive Steganography

Dr I. Kala – Associate Professor
Dept of Computer Science and Engineering,
PSG Institute of Technology and Applied Research,
Coimbatore - 641062
ootykala@gmail.com

Ms. Carol S
Dept of Computer Science and Engineering,
PSG Institute of Technology and Applied Research,
Coimbatore - 641062
11carol1102@gmail.com

Mr. Krishna Kumar D
Dept of Computer Science and Engineering,
PSG Institute of Technology and Applied Research,
Coimbatore - 641062
krishna24002@gmail.com

Ms. Nehya Ragavan V A
Dept of Computer Science and Engineering,
PSG Institute of Technology and Applied Research,
Coimbatore - 641062
rnehya222@gmail.com

***Abstract*—With the growing need for secure communication, steganography has emerged as a vital information security technique. However, traditional steganography methods face limitations in adaptability and content protection. This paper proposes a novel approach utilizing AI-powered adaptive steganography with dynamic content protection. The proposed system leverages artificial intelligence to dynamically adjust steganographic techniques based on the cover media format and potential threats. Furthermore, it integrates dynamic content protection mechanisms that allow for real-time modification of the hidden content based on predefined rules or user interactions. This paper details the design, implementation, and evaluation of the proposed system. The results demonstrate the effectiveness of AI-powered adaptation and dynamic content protection in enhancing steganographic security. Finally, the paper discusses the limitations, future directions, and potential applications of this innovative approach.**

***Index Terms*—Steganography, AI (Artificial Intelligence), Contrastive Learning, GAN Network.**

I. Introduction

The ever-increasing demand for secure communication has propelled steganography to the forefront of information security techniques. While traditional methods offer a layer of concealment, they often lack adaptability and robust content protection mechanisms. This paper presents a novel approach that bridges this gap by introducing AI-powered adaptive steganography. This innovative system leverages artificial intelligence to dynamically optimize steganographic techniques based on the cover media and potential threats. This paper delves into the design, implementation, and evaluation of this AI-powered steganography system, demonstrating its effectiveness in enhancing steganographic security for the modern digital landscape.

A. Introduction to AI in Steganography

Steganography, the practice of concealing secret information in seemingly commonplace media, has historically struggled with issues related to hiding ability, resilience, and medium adaptability. But there's a promising answer provided by the rapidly expanding science of artificial intelligence (AI), particularly deep learning. After being trained on enormous datasets, deep learning models are highly skilled at spotting complex correlations and patterns. These models can be used to assess cover media (photos, sounds, and videos) in the context of steganography with an unprecedented level of detail. This study reveals a wealth of information about the intrinsic qualities of the media, including statistical features and noise distribution.This information can be leveraged to develop sophisticated embedding algorithms that enhance steganographic efficacy and resilience. By integrating AI into steganography, several key challenges can be addressed:

1) **Enhanced Security:** AI-driven models can optimize imperceptibility and security by adaptively changing the embedding process according to the host media's properties. By producing stego media that are identical to their cover counterparts even when subjected to statistical analysis, these models can also negate the effects of steganalysis methods.

2) **Enhanced Capacity:** The payload capacity can be raised using neural networks without sacrificing the media's quality or detectability. This is accomplished by dynamically and intelligently modifying the embedding algorithms to take into account the media's spatial and frequency domain properties.

3) **Robustness Against Attacks:** Artificial intelligence (AI) can strengthen steganographic systems' resistance to a variety of attacks, including as noise addition, cropping, and compression.

B. Deep Learning For Content Analysis

Artificial intelligence's deep learning branch has become a potent instrument for content analysis across a range of industries, including steganography. Deep learning models are highly effective at automatically learning complicated representations from large amounts of data, in contrast to standard methods that rely on handcrafted features. If you were to display a deep learning model millions of photographs, it would be able to gradually understand all the fine information about the textures, colors, and forms of the objects in the images. Deep learning models can be trained on an enormous

dataset of cover media (pictures, audio, and videos) and their matching stego-media (media with hidden messages) in the context of steganographic content analysis.

Through this training, the model can recognize minute changes made throughout the message embedding process and comprehend the image's inherent properties. By examining these attributes, the deep learning model can virtually "fingerprint" the cover material, acquiring a thorough comprehension of its innate statistical traits and noise patterns. This comprehensive knowledge is essential for the following phase, known as adaptive message embedding, in which the AI uses it to conceal messages inside the cover medium while retaining imperceptibility strategically. By enabling a more reliable, flexible, and secure method of covert communication, deep learning is essential to realizing steganography's full potential.

This method transforms conventional steganography by letting the model select the best way to embed data based on the properties of the medium, rather than depending on preset embedding methods. Each instance of message embedding is distinct and customized to the particular media file thanks to the neural network's inherent capacity to assess and comprehend the media.

The model's ability to learn from start to finish greatly increases the effectiveness of this strategy. It can simultaneously learn the most effective methods for extracting and embedding concealed messages, resulting in a synchronized system that is optimized for both operations. The efficiency of steganographic systems is greatly increased by this dual capability, which also makes them faster and more dependable.

Furthermore, the application of Generative Adversarial Networks (GANs) in this field makes it easier to identify nuances that were missed by conventional techniques. Because GANs replicate the statistical characteristics of real cover media, they can be used to create synthetic stego-media that is extremely difficult for even the most advanced steganalysis algorithms to detect.

Deep learning can adapt to new media and growing digital environments, such 3D images and high-definition films, because of its capacity to learn from large datasets. Because of its flexibility, AI-driven steganography is guaranteed to continue working as digital media develops, offering a reliable solution that is resistant to shifts in media consumption and technology.

In summary, deep learning applied to steganography not only improves the conventional data hiding capabilities but also creates new opportunities for secure communications research and development. The potential for undetectable communication will increase as these models get more complex and datasets get bigger, which will be a huge advancement in the realm of information security.

II. PROJECT SCOPE AND MOTIVATION

The ongoing need for reliable and secure communication techniques, especially in situations when privacy is crucial, is what motivates our initiative. Although they provide a layer of secrecy, traditional steganography techniques have drawbacks such as low hiding capacity, assault susceptibility, and limited media adaptability. Their use in practical settings is limited by these drawbacks. Our initiative uses deep learning, a type of artificial intelligence (AI), to harness the revolutionary power of AI to address these issues. We provide a steganography system driven by AI that can dynamically safeguard stuff. Two major uses of deep learning models in this system are adaptive message embedding and content analysis. The initial deep learning model will be trained to examine the cover media in great detail. Preserving the secret message will be the main objective of the second deep learning model. In order to recognize and immediately thwart steganographic attacks, our model will continuously monitor the stego-media—that is, media that contains the hidden message. The successful creation of this AI-powered steganography system could greatly improve steganography's efficacy and security in a variety of contexts. The goal of this project is to develop a strong and flexible steganography method that may be used for covert data transmission in resource-constrained contexts, copyright protection for digital media content, and secure communication in privacy-critical scenarios.

III. PROPOSED WORK

A. Data Collection:

In order to train a strong AI for steganography, a broad approach to data collection is required. We'll compile a sizable collection of cover photos in different categories and formats. Additionally, we'll gather a variety of encrypted messages in picture format. We will also obtain stego-images produced using various methods in order to comprehend the best ways to incorporate data and avoid detection. In certain situations, it can be beneficial to include user input regarding how well-hidden the messages are in the pictures. In order to enable the AI to develop robust embedding strategies, we will finally compile knowledge on current techniques for detecting buried messages. The AI will be ready to excel at concealing information within photos thanks to this thorough data collection process.

B. Generating Embeddings

When it comes to steganography, text embedding creation is essential to safely and successfully integrating text into cover material. One popular approach to creating these embeddings is the Word2Vec model, a reliable method from natural language processing (NLP) that converts text to a machine-understandable numerical format.

Word2Vec is a predictive deep learning model that processes text using either Skip-gram or Continuous Bag of Words (CBOW) architectures. It basically learns to anticipate words either from their context (Skip-gram) or from words themselves (CBOW). Dense, fixed-size, and semantically relevant vector representations of words are produced in a high-dimensional space by this prediction process.

We use these embeddings to encode the secret messages that must remain concealed within the cover media in order to perform steganography. The following steps are involved in the process:

1) Training or Loading the Word2Vec Model: We can either utilize a pre-trained model or create a Word2Vec model from scratch utilizing a sizable corpus of text pertinent to the communication's context. Large volumes of text data are fed into the model during training so that it can acquire a vocabulary and the embeddings of each word.

2) Text Processing: The secret message is broken down into words, or tokens, and then the Word2Vec model is used to convert each token into its matching vector. In this step, the written message is converted into a series of vectors, making it ready to be embedded into digital media.

3) Creating Text Embeddings: A single, cohesive embedding for the complete text can be created by processing the individual word vectors further. This can be accomplished in a number of ways, such as by averaging the vectors, which makes the embedding simpler while keeping the message's core semantic properties. Alternatively, to minimize dimensionality and capture deeper semantic meanings, more sophisticated methods such as Principal Component Analysis (PCA) or an autoencoder might be employed.

Steganography can benefit from embeddings that are not just dense and information-rich but also customized to fit in with different kinds of digital material through the use of Word2Vec and later processing techniques. By improving the steganographic process' efficiency and security, this technique makes it very difficult for unauthorized parties to figure out whether any hidden messages are there.

C. LSB Steganography

The technique known as Least Significant Bit (LSB) steganography is extensively used to embed confidential data, such text embeddings, into digital photographs while causing the least amount of visual degradation possible. By modifying the least important bits of these values to encode the data, this technique takes advantage of the binary representation of pixel values in an image. The embedding of text-embeddings produced by models such as Word2Vec must first be transformed into a binary sequence. Next, this sequence is deftly threaded across the image into the least important pixel values. The main benefits of LSB steganography are its high level of imperceptibility and ease of use.

D. Generative Adversarial Network

Generative Adversarial Networks (GANs) can be used to further improve AI-powered steganography and produce a dynamic content protection solution. Imagine a system that could both modify the material to fit the message and conceal hidden messages within photos. This is where GANs are useful. Two neural networks engaged in an adversarial training conflict make up a GAN. The hidden message is included in fresh images (stego-images) made by one network, the generator. The discriminator, the other network, attempts to distinguish these stego-images from the original cover images. The discriminator is compelled to improve its detection skills as the generator becomes more adept at producing undetectable stego-images. The constant competition forces the GAN to create extremely flexible steganography methods.

The generator is able to learn how to subtly alter the background color or texture of the cover image in order to smoothly incorporate the message without drawing attention to itself.By blending the embedded data smoothly with the cover medium, this procedure guarantees that it is virtually undetectable to the naked eye and even to some automated analysis methods. By continuously improving, the generator pushes the limits of steganographic approaches by precisely targeting weaknesses in the discriminator's detection capabilities. The embedding and detection processes benefit from each other's enhancement in this cat-and-mouse dynamic, which improves the security and effectiveness of data concealing in digital media.

IV. SYSTEM DESIGN

A. Data Acquisition And Preprocessing:

The quality and preparation of the data are critical to the system's success. The steganography technique is powered by two main data sources: cover photos and hidden messages. The cover photos serve as the concealed messages' imperceptible containers. The way the system gets them may include a number of different approaches. The model could make use of publicly accessible datasets that include a variety of image categories, such as objects, landscapes, and portraits, in order to guarantee that it generalizes effectively across various image attributes. Because cover images might differ significantly in real-world circumstances, this diversity aids in the model's adaptation. As an alternative, users may be able to contribute their own cover photos, albeit possibly with limitations on size and format to make sure they work with the pipeline for processing.

Depending on how the system is designed, the information to be embedded—secret messages—can take many different forms. Text messages are a popular choice, but they need to be converted into numerical representations that neural network processing can understand. This makes it possible for the model to efficiently comprehend and work with the text data. The system may require conversion of audio messages to a particular format, such as spectrograms, which graphically display the frequency content of the audio. To maintain uniformity across the training set, the system may also normalize the audio length. It is possible to allow image messages as well, but in order to maintain compliance with the cover image format, they may need to be resized or converted to grayscale. This preprocessing stage makes sure that all message formats are given in a way that is easy for the model to comprehend and work with the cover photos.

Following acquisition, preprocessing procedures are used to both cover images and hidden messages in order to get them ready for GAN model training. It may be necessary to resize cover images to a consistent size that fits the neural network design. Because of this standardization, the model can scan all images quickly and concentrate on figuring out the underlying connections between cover images and the messages they hide. Techniques for image augmentation such as flipping, random

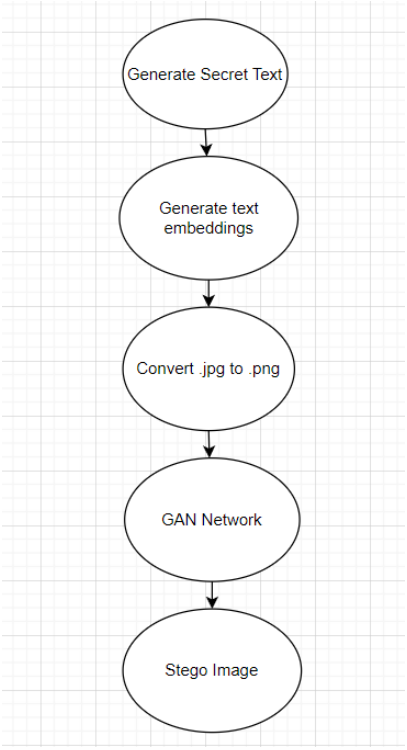


Fig. 1. Flow Diagram

cropping, and color jittering can also be used. By producing variants of already-existing photos, these strategies artificially increase the size of the dataset. This makes the model more resilient to changes in real-world image quality, enhancing its capacity to manage hidden data in steganography jobs. Secret message preprocessing is contingent upon message format. While audio or image signals may go through format conversion or normalization, text messages may be transformed into numerical vectors. By ensuring that all data is delivered in a standardized, machine-readable format, these pretreatment procedures enable the GAN model to efficiently pick up the skills necessary for adaptive steganography.

B. Model Architecture Design:

The GAN architecture comprises two primary components: the generator and the discriminator, each playing a crucial role in the embedding and detection of hidden information within digital media.

Generator Architecture: The generator is designed to synthesize stego-images that incorporate hidden information seamlessly into cover images. It takes two inputs: an image and its corresponding text-embedding. The text embedding is first reshaped to match the dimensions of the input image, allowing for a unified processing pathway. This is achieved by reshaping the embedding into a 1×1 $1{\times}1$ spatial configuration with depth equal to the embedding size, followed by padding to extend its dimensions to match those of the input image. The padded embedding is then concatenated with the input image, creating a combined feature map.

A sequence of convolutional and transposed convolutional layers process the concatenated map. Conv2D layers with LeakyReLU activation and dropout are used by the network for regularization, which improves generalization and guards against overfitting. After every convolutional layer, batch normalization is used to stabilize learning and accelerate convergence. The end result is a Conv2DTranspose layer, which embeds the concealed message in the image and reconstructs it to its original dimensions.

Discriminator Architecture: Making the distinction between stego-images and cover images is the discriminator's job. The text embedding is altered and combined with the image in a similar input procedure. To further improve discrimination capabilities, the discriminator employs a sequence of Conv2D layers with LeakyReLU activation, each followed by dropout. The final convolutional layer's output is flattened by the network, which then runs it through a dense layer with sigmoid activation to provide a validity score that represents the probability that the input is a stego-image.

Integration of Adversarial Models: By creating progressively similar stego-images, the generator seeks to trick the discriminator in a training scenario where the adversarial model integrates the discriminator and generator. To make sure that no component is impacted by the other's continuing learning while both are learning at appropriate levels, the discriminator's weights are frozen throughout the training phase and vice versa.

Learning and Optimization: Adam optimizers with learning rate schedules determined by exponential decay are installed in both the discriminator and the generator. As the networks gain more proficiency in generating and discriminating signals, this configuration aids in adjusting the learning rates during training to the task's complexity.

By effectively learning the art of steganography and modifying its embedding approaches to suit different media types and embedding obstacles, the system's architecture and training strategy work together to push the frontiers of covert communication through digital media.

C. Deep Learning Training Pipeline:

The process of turning raw data into a potent AI model for adaptive steganography is orchestrated by the deep learning training pipeline. The pipeline first gathers and prepares the data, loading cover images that have already been processed, hidden messages, if any, and pre-generated embeddings that effectively capture both. There may be more framework-specific preprocessing involved, such as data normalization. Subsequently, the pipeline builds each generator and discriminator network according to the pre-established design. After that, these networks are assembled, with each network's loss function (which directs performance assessment) and optimizer (which modifies internal parameters during training) specified.

The discriminator and generator are repeatedly trained over several cycles in the adversarial training loop, which is the central component. By examining real cover images and their

embeddings, the discriminator improves its capacity to discern between real and phony images inside each cycle. In turn, the generator gains the ability to produce stego-images that deceive the discriminator. This fosters a competitive atmosphere in which both networks advance steadily. The pipeline then assesses the model's performance post-training, taking into account things like the robustness of the model against steganalysis approaches and the imperceptibility of stego-images. In order to maximize its steganography capabilities, the pipeline may modify the model architecture or training parameters in light of these findings.

D. Content Adaptation Strategies:

This system's capacity to dynamically modify cover picture content during steganography is one of its primary features. This modification makes sure that the secret message is successfully embedded while yet preserving the image's authentic aspect. The system may use a few different approaches to accomplish this. Finding the parts of the cover photo that have the most room for editing while maintaining the least amount of visual impact is one method. These could be spaces with few textures, subtle color changes, or little detail. Then, algorithms can slightly modify these areas according to the embedded message. To accommodate the message, the alterations may entail altering the textures, colors, or pixel values inside the selected locations, all while making sure the changes are imperceptible to the naked eye.

An alternative approach makes use of the message itself to direct the process of content adaptation. The message's attributes may be examined by the system, which would then utilize this data to decide what changes to make to the cover image's content. For example, incorporating a message with a high contrast image may need adjusting the cover image's parts with greater contrast. In order to achieve a high degree of imperceptibility—that is, to make it difficult to discern whether a hidden message exists inside the stego-image—these content adaption tactics are essential. The method may effectively hide secret messages while preserving the image's natural appearance by carefully altering certain parts of the cover image.

E. Message Embedding:

The secret messages can be effortlessly embedded into cover photos, which is the primary functionality of this system. The trained Generative Adversarial Network (GAN) model is utilized in this process. Numerical representations of text messages that are appropriate for neural network processing may be created. This makes it possible for the GAN model to efficiently interpret and work with the text data. For consistency within the model, audio messages may need to be normalized to a specified audio length and converted to a particular format, such as spectrograms. Image messages may also be supported; to make sure they work with the cover image format, they may be resized or converted to grayscale.

The process of creating an embedding involves training a different model (such as an autoencoder) to learn compressed representations, or embeddings, that capture the essence of the message data. Essentially, embeddings serve as a clear and informative way to represent the message for the GAN model. The specific technique for embedding generation may vary depending on the chosen implementation.

GAN Integration: The message embedding and the preprocessed cover image are ready, and the GAN enters center stage. The generator network receives the cover picture and its matching embedding. The generator modifies the cover image to produce a stego-image, led by the message embedding. The goal of this alteration procedure is to preserve the image's natural appearance while carefully altering certain content within the cover image to convey the message. The effectiveness of the GAN model, which has been trained using the adversarial training pipeline to be proficient at this particular task, is what will determine whether or not this integration is successful.

Stego-Image Generation: The end result of this procedure is a cover image known as a stego-image, which obstructs the embedded message while appearing to be unaltered to the unaided eye. The concealed message within its digital information can then be carried by transmitting or storing this stego-image. One critical component of the system's effectiveness is the stego-image's imperceptibility and message integrity.

F. Security And Robustness Considerations:

It is crucial to guarantee the confidentiality and resilience of concealed communications in this steganography system. There are two main points that are discussed: resistance to steganalysis and imperceptibility. The term imperceptibility describes how difficult it is to discern whether a hidden message exists within a stego-image. The system uses a few different strategies to do this. The tactics for content adaptation are quite important; they involve carefully altering some parts of the cover image in order to incorporate the message while reducing its visual effect. Furthermore, the GAN model is trained to produce stego-images that bear a striking resemblance to the original cover photos. To evaluate the imperceptibility of the stego-images, measures such as human evaluation studies or Peak Signal-to-Noise Ratio (PSNR) can be employed.

Steganalysis resistance is still another important security factor. The term "steganalysis" describes methods for looking for hidden messages in pictures. The system employs multiple tactics to thwart efforts of steganalysis. Here, the GAN model's strong training is crucial. The model makes it harder for steganalysis algorithms to find anomalies in the stego-images by continuously enhancing its capacity to produce realistic stego-images during the adversarial training phase. To offer an additional degree of protection in the event that the stego-image is hacked, the system may also use encryption techniques for the message before embedding. The approach aims to establish a strong and secure communication channel for sending secret messages by emphasizing both imperceptibility and resistance to steganalysis.

G. Evaluation Metrics:

To make sure that this adaptive steganography system hides information efficiently without sacrificing image quality, it is imperative to assess its performance.

Imperceptibility: This measure evaluates the degree to which the stego-image conceals the underlying message. Methods such as Peak Signal-to-Noise Ratio (PSNR) are frequently applied. Higher PSNR values indicate better imperceptibility since they indicate greater resemblance between the original cover picture and the stego-image. PSNR assesses this similarity. Furthermore, research including human evaluation can be carried out in which individuals are asked to differentiate between original and stego-images, thereby offering a subjective evaluation of imperceptibility.

Message Embedding Success Rate: This indicator shows how well the system was able to include the hidden message into the cover photo. It can be computed as the proportion of messages that a message extraction method correctly recovers from the stego-image. To guarantee the integrity of the covert communication, a high message embedding success rate is necessary.

Robustness Against Steganalysis: This measure assesses how well the system withstands methods designed to find hidden messages. To determine how challenging it is to distinguish stego-images from actual cover photos, the system may be put to the test using a variety of steganalysis techniques. The robustness of the system can be measured using steganalysis technique-specific metrics or by looking at the success rate of the steganalysis algorithms themselves. The system's creators can learn a great deal about the system's advantages and disadvantages by examining these measurements. To increase the steganography system's overall efficacy, this data can be utilized to improve the model architecture, training parameters, or content adaption techniques.

V. IMPLEMENTATION OF THE FRAMEWORK

A. Setting Up Environment:

The project creates a deep learning environment for the AI model's training by utilizing Python and TensorFlow/Keras. There is no need for local setup with Google Colab's preconfigured environment, which comes with crucial libraries like TensorFlow and Keras already installed. It also offers free usage of strong GPUs, which are essential for effectively training the deep learning models that are employed in this research. To utilize Colab, just import the necessary libraries into a brand-new Jupyter Notebook that you create on the platform. After that, the notebook serves as the development environment where the model architecture is defined, data is loaded, and the AI system for adaptive steganography is trained.

B. Data Preprocessing:

Preparing the data is an important step since it makes sure the data is in a format that the model can be trained on. Several preprocessing processes were performed in the context of our lane detection system in order to get the input data ready for training.

1) Cover Image Dataset Acquisition: Obtaining a broad dataset of cover photos is the initial stage. A diverse range of image categories, such as objects, landscapes, and portraits, should be included in this dataset to guarantee that the system can effectively embed messages across various image attributes and generalize well. Selection criteria could include things like color depth, image resolution, and the lack of obvious artifacts or noise that could impede the steganography process.

2) Cover Image Preprocessing: Preprocessing methods are used to get the cover image dataset and get the photos ready for GAN training. Resizing every image to a standard dimension appropriate for the neural network design may be necessary to achieve this. Furthermore, to artificially increase the dataset and strengthen the model's resistance to changes in real-world photos, image augmentation methods like random cropping, flipping, or color jittering can be used.

3) Image Format Conversion: Standardizing the input picture formats is a crucial step in getting data ready for steganographic operations. Images in datasets typically come in a variety of formats, the most popular ones being JPEG (.jpg) and PNG (.png). Because PNG is a lossless format and is essential for maintaining the integrity of embedded data in steganography, we must convert all images to this format. The conversion procedure is simple but very important. Reading the original JPEG files, decoding them into unprocessed pixel data, and then encoding this data into the PNG format are the steps involved. This guarantees that the pixel data is not changed during the steganography encoding and decoding operation, protecting the concealed message. The following is a quick rundown of the technical processes involved:

1) Reading the JPEG File: After a JPEG image is imported into the system, the compressed data is decoded and the image is represented as a grid of pixels in a bitmap format.

2) Conversion to PNG Format: Next, the bitmap is reencoded into PNG format after it has been decoded. In this phase, the bitmap data is compressed using the lossless compression technique of PNG, which guarantees the preservation of all pixel information.

3)Keeping the PNG Image: After it has been made, the PNG image is stored to a specified directory so that it can be used for additional steganographic processing. To ensure that all images utilized in the training and application of the GAN for steganography are in the PNG format and meet the requirements for lossless data handling in embedding secretive information, this preprocessing phase is included into the larger system.

C. Text and Embeddings Generation

Effective generation and processing of textual data is essential in the field of steganography, especially when the cover medium is an image. The creation of text and its subsequent transformation into numerical embeddings—which are easily incorporated into images—are the main topics of this section.

We use the powerful Word2Vec model from natural language processing to convert text to a numerical format that can be processed by deep learning models.

Text Generation: To mimic the embedding of different messages into photographs, the system generates textual data. Using a preset dataset—which may contain common terms or carefully produced messages meant for embedding—randomly selecting words or phrases is what this technique entails.The diversity of the generated text guarantees the system's resilience and enables it to manage practical situations in which the embedded text varies greatly. Word2Vec for Text Embedding: The Word2Vec model is used to transform text into embeddings once it has been generated. Word2Vec is a predictive deep learning model that turns words into high-dimensional vectors using either the Continuous Bag of Words (CBOW) or Skip-gram architecture. These vectors represent semantic meanings in a way that places words in the corpus that have similar contexts close to each other. When a Word2Vec model is trained, it involves:

1) Corpus Preparation: The Word2Vec model must be trained on a sizable corpus of text. This corpus may be generic or domain-specific, depending on how the steganography system is supposed to be used.

2) Training Models: The Word2Vec model is designed to predict words either by their context (CBOW) or by the word itself (Skip-gram). The word embeddings, or internal weights, of the model are adjusted by use of optimization methods and backpropagation.

3) Embedding Extraction: The trained Word2Vec model creates a vector in a multi-dimensional space for every word in the resulting text. The embedding is this vector (or series of vectors, for phrases or sentences).

Application to Steganography: The resulting embeddings are subsequently fed into the steganographic apparatus. To ensure that the embedded information is both recoverable and undetectable, these embeddings can be modulated or concatenated with image data in the context of an image steganography GAN.

This method of producing text and its embeddings enables a versatile steganography system that can accommodate different kinds of concealed messages while preserving the resilience required for efficient concealment and later retrieval of the embedded data.

1) Data Pairing and Splitting: The data must be paired after the cover photos, hidden messages, and related embeddings have been preprocessed. This entails forming pairs, each of which has a secret message embedding and a cover image that has been previously processed. Ultimately, the data is divided into sets for training, validation, and maybe testing. The GAN model is trained on the training set, monitored throughout training by the validation set, and, if applicable, tested on unseen data to assess the performance of the final trained model.

VI. MODULE WORKFLOW

This project's primary goal is to produce adaptive steganography by training a deep learning model—more precisely, a Generative Adversarial Network, or GAN.

1) Model Architecture Definition: This first stage entails defining the GAN model's architecture. A discriminator and a generator are the two sub-networks that make up the model. Stego-images—images that subtly incorporate secret messages—are produced by the generator network. Its architecture probably makes use of convolutional layers to efficiently process picture data, and it might also make use of methods like transposed convolutions to produce new images. As a critic, the discriminator seeks to distinguish between the generated stego-images and the actual cover images. It may have a generator-like architecture, but it focuses on learning traits that differentiate authentic from fraudulent photos. This step defines the specifics of the architecture, such as the quantity, kind, and arrangement of connections between the layers in each network.

2) Data Loading and Preprocessing: The module concentrates on loading and preparing the data before training the GAN. This entails loading, depending on the implementation, the preprocessed cover picture dataset as well as maybe the preprocessed secret message dataset. Pre-generated embeddings for cover photos and secret messages may also be included in the data. These compressed representations, known as embeddings, are essential to the steganography process because they capture the substance of the material. Here, additional preparation actions, including data normalization or conversion to particular data types, could be used that are particular to the deep learning framework of choice (TensorFlow/Keras).

3) Model Building and Compilation: This step is all about constructing the different parts of the GAN after the model architecture has been established and the data has been prepared. The code outlines how to build the generator and discriminator networks using the previously chosen architecture as a basis. Within the selected deep learning framework, these functions convert the architectural design into functional neural network models. The models are then assembled using the proper optimizers and loss functions. The optimizer directs the training process by changing the internal parameters of the model to minimize the loss, while the loss function aids the model in understanding how well it is working.

4) Adversarial Training Loop: The core of the training procedure is this. The generator and discriminator are trained iteratively over several epochs (full training cycles) in a training loop that is implemented by the code. The discriminator is trained initially by the loop during each iteration. To enable the discriminator to learn to discriminate between genuine and created data, it is fed real cover images together with their appropriate embeddings. The generator is subsequently trained. It aims to create stego-images that can trick the discriminator by using the embeddings of real cover pictures that it receives. An antagonistic training dynamic is produced

as a result. While the generator works to produce stego-pictures that are more realistic and challenging to differentiate from actual images, the discriminator is continuously refining its capacity to identify fraudulent images.

5) Model Evaluation and Refinement: The model's performance is assessed following a predetermined number of epochs of GAN training. This could entail evaluating the stego-images' imperceptibility, or how successfully the hidden signals are concealed inside them. For this, methods such as the Peak Signal-to-Noise Ratio (PSNR) or human evaluation studies can be employed. It might also be assessed how resilient the model is to steganalysis tools (which find hidden messages). To enhance the overall performance of the model in terms of imperceptibility, robustness, and maybe the quality of the generated stego-images, the evaluation results may lead to adjustments being made to the model architecture or training hyperparameters.

A. MODULE IMPLEMENTATION

1) Defining Model Architectures : The steganographic system's architecture is built around a Generative Adversarial Network (GAN) framework, which is intended to integrate and mask text embeddings in images. The architectural elements of the generator and discriminator models—which are critical to the embedding and identification of hidden information—are described in this section.

Generator Architecture: The main function of the generator is to create stego-images, or images with text contained in them, that are identical to regular, non-stego images. Its two inputs are an image and a text embedding, from which it generates an altered image with the text embedded in it.

Layers of Input: Image Input: An RGB standard picture with the shape (224,224,3) is provided to the generator.

1) Text Embedding Input: The generator is given a condensed representation of the text to be embedded in the form of a text embedding vector of shape (1,128) together with the image.

2) Embedding Reshaping and Concatenation: A ZeroPadding2D layer is used to extend the text embedding throughout the input image's spatial dimensions after it has been initially reshaped using a Reshape layer to match the image's spatial dimensions.The textual and visual data are then combined at the pixel level by concatenating the altered embedding with the picture input.

3) Convolutional Layers: Several convolutional layers are applied to the combined output. The filters applied by each convolutional layer, more especially the Conv2D layers, integrate and capture the spatial hierarchy between the image features and the text embedding. After every convolution, batch normalization and LeakyReLU activation functions are applied to bring non-linearity and stabilize the learning process.

4) Transposed Convolutional Layers: Conv2DTranspose layers are applied after the convolutional layers in order to upsample the feature maps back to the original picture size. This effectively embeds the text into the image while attempting to preserve the original image's visual quality.

5) Layer of Output: Conv2DTranspose, the last layer, produces a three-channel RGB image that is the same size as the input image and now has embedded text in it.

Discriminator Architecture: The discriminator's job is to distinguish between the original, text-free images and the stego-images that the generator produced.

1) Input Layers: The discriminator gets an image and a text embedding, which are preprocessed similarly to the generator to match their dimensions. Convolutional processing and concatenation: Several convolutional layers are applied to the combined input following concatenation. LeakyReLU activation and dropout are used in each layer for regularization, which improves the discriminator's capacity to identify inconsistencies brought about by the embedding procedure. 2) Flattening and Output: A dense layer with a sigmoid activation function processes the one-dimensional vector output from the final convolutional layer, classifying the input as a real or altered image.

Adversarial Model Integration: The adversarial model integrates the discriminator and generator in a training loop in which the discriminator gains more accuracy in detecting progressively subtle stego-images that the generator attempts to mislead it into detecting. By ensuring that the generator learns how to embed text into images in a virtually undetectable manner, this training mechanism improves the steganographic method's security and robustness.

2) Data Loading and Preprocessing Functions: Writing routines to manage data loading and preparation chores is part of this module. It may be possible to create functions that load the preprocessed cover picture dataset from predetermined locations. This could contain any additional preprocessing unique to the deep learning system, such as data standardization or conversion to tensors. Functions that load pre-generated embeddings for both cover pictures and secret messages, as well as preprocessed secret message data (if applicable), may also be included, depending on the implementation. These features make sure the model receives the data in a way that is appropriate for training.

Transforming JPG image files into PNG image files is a crucial step in the steganography system's data preprocessing phase. This conversion is noteworthy for a number of reasons:

1) Lossless Compression: PNG employs lossless compression, which is essential for preserving the integrity of the image since no image data is lost during the storing process. Because every pixel in steganography has the ability to contain small amounts of hidden information, this aspect is especially crucial. 2) Support for Transparency: PNG, as opposed to JPG, has transparency support, which is useful for some steganography-related image processing activities, particularly those that call for layering or other transparency-requiring picture manipulation.

3) Better Text and Sharp Edge Quality: PNG is known to handle text and sharp edges better than JPG, which frequently exhibits artifacts around these regions and uses lossy compression. Because steganography may need fine-grained adjustments to text or image edges, PNG's ability to manage

these components efficiently is advantageous. Procedure of Conversion:

The PIL (Pillow) package in Python is used to implement the JPG to PNG conversion. It offers an extensive collection of image processing functions. Below is a condensed description of the conversion's steps:

1)Load the picture: Pillow uses the Image.open() function to load each JPG picture into memory.

2)Convert the Image: JPG images are saved directly in PNG format by using the save() method with the PNG format provided, even if Pillow automatically decodes them into a raw format. This technique guarantees lossless compression during image storage.

3)Preserve Image Quality: The original visual information in the image is kept as much as possible by not applying any further compression or quality adjustments during the saving process.

Usually, this conversion phase is carried out in a batch procedure that converts several photos sequentially.

3) Model Compilation with Loss Functions and Optimizers: The compilation of the separately defined generator and discriminator models is the main goal here. With the help of Keras' model compilation features, you may define an optimizer and a loss function for every network. The loss function aids in the model's performance evaluation during training. A loss function (such as mean squared error for picture similarity) that promotes the creation of realistic stego-images might be applied to the generator. Alternatively, the discriminator's loss function (binary cross-entropy, for example) might be built to penalize it for misclassifying real or fraudulent photos. Furthermore, you will designate an optimizer (such as the Adam optimizer) that directs the training procedure by modifying the networks' weights and biases in accordance with the determined loss values.

4) Implementing the Adversarial Training Loop: The adversarial training idea is translated into code in this part. The training data, which consists of cover images and their embeddings, is iterated through several times using a loop. The loop trains the discriminator first in each iteration. The discriminator is fed actual cover photos together with their embeddings, and the optimizer computes the loss and applies it to the discriminator's internal parameters based on the discriminator's predictions. The generator is subsequently trained. The generator is fed actual cover images together with their embeddings by the loop, which instructs it to produce stego-images.

The discriminator is then given these stego-images along with the cover image embeddings that correlate to them. However, this time the gradients are utilized to modify the generator's parameters, so in effect rewarding the generator for producing stego-images that can deceive the discriminator. This loop powers the adversarial training process and lasts for a predetermined number of epochs.

5) Model Saving and Loading: It is imperative to store the trained GAN model for later usage. The trained model architecture, weights, and optimizer state can all be saved to disk using Keras' features. This enables you to evaluate the trained model's performance on fresh data or load it later for purposes like creating stego-images. Depending on your deep learning framework of choice and the demands of the project, the particular file format (such as HDF5) that is utilized to store the model may change.

6) Generating Stego-Images: This module contains functionality for exploiting the trained GAN model to generate stego-images, depending on the scope of the project. This may entail a function that takes in the trained generator model and an actual cover image with its embedding, and outputs a stego-image that embeds the hidden message in the cover image. The created stego-images may be subjected to post-processing methods (such as scaling pixels back to the original image range) prior to disk storage.

VII. EXPERIMENTAL SETUP AND RESULTS

The setting, resources, and techniques used in the creation and assessment of the Google Colab lane identification framework are described in the experimental setup. The experimental setting for our study is described below:

A. Hardware Configuration

GPU: NVIDIA RTX 6000 Ada Generation Graphics Card CPU:Intel® Xeon® Gold 6442Y Processor RAM:12GB Clock Speed: 2.60 GHz

B. Software Environment

The complex processing and computational requirements of deep learning models were supported by the stable software environment in which this project was built and carried out. The main elements of the software environment in use are listed below:

Operating System: Windows 11 was the operating system used for the development and training.

Programming Language: Because of its broad library support and simplicity in integrating with different data processing and machine learning frameworks, Python 3.8 was utilized for all scripting and model construction.

Deep Learning Framework: The foundation for creating and honing the neural network models was provided by TensorFlow 2.4 and Keras 2.4.3. While TensorFlow delivered the frameworks and tools required to effectively develop complicated models, Keras supplied a high-level API that expedited prototyping.

- NumPy: Used for high-performance numerical computing and data manipulation.
- OpenCV: Employed for image processing tasks, including reading, writing, and transforming images necessary for data preprocessing.
- Matplotlib: Utilized for creating visualizations of training metrics and results to better interpret the model's performance.
- Pandas: Used for efficient data manipulation and analysis.
- Scikit-Learn: Leveraged for preprocessing data and evaluating model performance through various metrics.

Training Set			
TARGET / OUTPUT	Positive(P)	Negative(N)	SUM
Positive(P)	25479 48.08%	1017 1.92%	26496 96.16% 3.84%
Negative(N)	8742 16.50%	17754 33.50%	26496 67.01% 32.99%
SUM	34221 74.45% 25.55%	18771 94.58% 5.42%	43233 / 52992 81.58% 18.42%

Fig. 2. Confusion Matrix

- Version Control: Git was used for version control, with the repository hosted on GitHub to maintain a record of all changes and enable collaborative development.

C. Dataset

The Oxford Flowers 102 dataset from Kaggle is chosen as the dataset for this project.This dataset contains 8,189 images of flowers categorized into 102 classes, with each class representing a type of flower. This dataset is particularly is chosen because it offers a large, yet manageable number of images in a single category (flowers), making it ideal for the training models to focus on subtle distinctions.

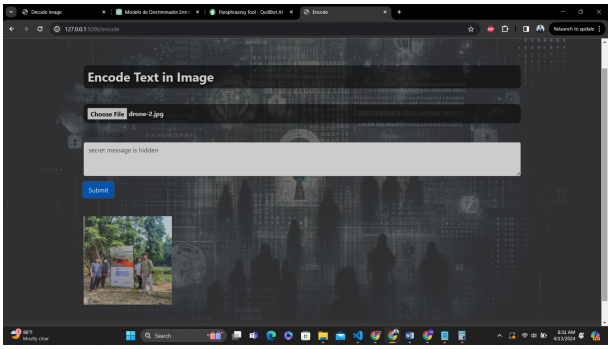
D. Model Training

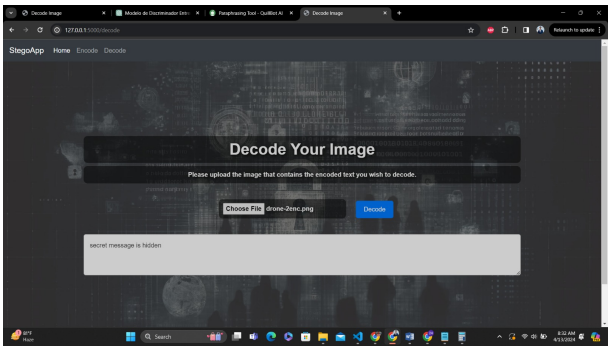
A. Architecture The generator is structured as a convolutional neural network (CNN) composed of both convolutional and deconvolutional layers. The input to the model comprises two distinct elements: the image and the corresponding text embedding. The text embedding, shaped as a 1x1x128 vector, undergoes reshaping and zero-padding to align with the spatial dimensions of the image input. Following this, the image and text embedding are concatenated and passed through several layers of the network: Convolutional Layers: The concatenated input is processed through Conv2D layers with LeakyReLU activation and BatchNormalization, followed by Dropout for regularization.

Deconvolutional Layers: Conv2DTranspose layers are utilized to upsample the feature maps and construct the steganographic output image. Each of these layers includes BatchNormalization and LeakyReLU activation, concluding with a sigmoid activation in the final layer to produce the image.

B. Training Procedure The generator is trained as part of the adversarial framework in conjunction with a discriminator model. The discriminator is responsible for distinguishing between real and generated images, providing the adversarial signal for the generator's training. The training procedure is as follows:

Loss Function: Binary cross-entropy chosen as the loss function, indicative of the adversarial game between the generator and discriminator.





Optimization: Adam optimizer with an exponential decay learning rate schedule is used. The generator's parameters are updated based on both the discriminator's feedback and the need to minimize the loss function.

Training Dynamics: During training, we alternate between updating the discriminator with real and generated images and training the generator to fool the discriminator. This involves feeding the discriminator batches of real images with text and generated images with the corresponding embeddings. The generator's weights are updated to maximize the confusion in the discriminator.

C. Hyperparameters

Batch Size: Set to 20 to allow for a detailed gradient update per training instance.

Epochs: Training is conducted for a l00 epochs

E. Model Evaluation

Generator Model: Generator Accuracy: 0.96 Discriminator Accuracy: 0.97

Decoder Model: Decoder Accuracy:0.95 Precision:0.9458206808374621 Recall:0.7600634057971014 F1 Score: 0.845

F. Performance Metrics

Generator Loss:0.135 Discriminator Loss:0.234 Generator Accuracy: 0.96 Discriminator Accuracy: 0.97 F1 Score: 0.845

VIII. CONCLUSION AND FUTURE SCOPE

In this project, we have successfully implemented a pipeline for embedding text into images using deep learning models. Starting with a dataset of images, we extracted features and

used a contrastive model to create embeddings. These embeddings were then fed into a GAN network along with a set of images to generate stego images with embedded text. Through our experimentation, we have demonstrated the feasibility of using neural networks to embed text into images, opening up new possibilities for data hiding and information security.

Moving forward, after creating a basic framework, there are a number of attractive directions that could be chosen, such as expanding message formats to include sensor and video data, improving content adaptation strategies with image manipulation techniques, encrypting messages more securely, investigating explainable AI in steganography for transparency, and identifying practical uses like digital watermarking and secure communication. By expanding the possibilities of covert message transmission, these developments aim to improve security and adaptability.

IX. References

[1] N. Subramanian, I. Cheheb, O. Elharrouss, S. Al-Maadeed, and A. Bouridane, "End-toEnd Image Steganography Using Deep Convolutional Autoencoders," IEEE Access, vol. 9, pp. 135585–135593, 2021, doi: 10.1109/ACCESS.2021.3113953.

[2] Y. Bhavani, P. Kamakshi, E. Kavya Sri, and Y. Sindhu Sai, "A Survey on Image Steganography Techniques Using Least Significant Bit," Lecture Notes on Data Engineering and Communications Technologies, vol. 101, pp. 281–290, 2022, doi: 10.1007/978-981-16-7610-920/COVER.

[3] A. Menendez-Ortiz, C. Feregrino-Uribe, R. Hasimoto-Beltran, and J. J. GarciaHernandez, "A Survey on Reversible Watermarking for Multimedia Content: A Robustness Overview," IEEE Access, vol. 7, pp. 132662–132681, 2019, doi: 10.1109/ACCESS.2019.2940972.

[4] J. Lu, W. Zhang, Z. Deng, S. Zhang, Y. Chang, and X. Liu, "Research on information steganography based on network data stream," Neural Computing and Applications 2020 33:3, vol. 33, no. 3, pp. 851–866, Aug. 2020, doi: 10.1007/S00521-020-05260-4.

[5] M. Hussain, A. W. A. Wahab, Y. I. bin Idris, A. T. S. Ho, and K. H. Jung, "Image steganography in spatial domain: A survey," Signal Processing: Image Communication, vol. 65, pp. 46–66, Jul. 2018, doi: 10.1016/J.IMAGE.2018.03.012.

[6] R. Hu and S. Xiang, "CNN prediction based reversible data hiding," IEEE Signal Processing Letters, vol. 28, pp. 464–468, 2021, doi: 10.1109/LSP.2021.3059202.

[7] T. Luo, G. Jiang, M. Yu, C. Zhong, H. Xu, and Z. Pan, "Convolutional neural networksbased stereo image reversible data hiding method," Journal of Visual Communication and Image Representation

[8] C. Yuan, H. Wang, P. He, J. Luo, and B. Li, "GAN-based image steganography for enhancing security via adversarial attack and pixel-wise deep fusion," Multimedia Tools and Applications 2022 81:5, vol. 81, no. 5, pp. 6681–6701, Jan. 2022, doi: 10.1007/S11042-021-11778-Z

[9] D. Li, Y. Yu, R. Ji, and H. Li, "Deep Learning for Digital Steganalysis: A Comprehensive Survey," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 50, no. 4, pp. 1403-1417, April 2020, doi: 10.1109/TSMC.2018.2876202.

[10] M. Barni, F. Perez-Gonzalez, "Coping with the Enemy: Advances in Adversary-Aware Signal Processing," in IEEE Signal Processing Magazine, vol. 37, no. 1, pp. 31-45, Jan. 2020, doi: 10.1109/MSP.2019.2949038.

[11] E. J. R. Alves, V. O. F. Lopes, A. N. Marana, "Neural Network-based Steganalysis in Digital Images," Journal of Information Security and Applications, vol. 50, pp. 102421, June 2020, doi: 10.1016/j.jisa.2019.102421.

[12] S. Lyu, H. Farid, "How realistic is photorealistic?", IEEE Transactions on Signal Processing, vol. 53, no. 2, pp. 845-850, Feb. 2005, doi: 10.1109/TSP.2004.840797.

[13] K. A. Shelby, R. C. Wade, "Steganography: Hiding Data within Data," IEEE Internet Computing, vol. 5, no. 3, pp. 75-80, May-June 2001, doi: 10.1109/4236.935182.

[14] G. K. Wallace, "The JPEG still picture compression standard," IEEE Transactions on Consumer Electronics, vol. 38, no. 1, pp. xviii-xxxiv, Feb. 1992, doi: 10.1109/30.125072.

[15] P. Sallee, "Model-based steganography," International Workshop on Digital Watermarking, pp. 154-167, Oct. 2003, doi: 10.1007/978-3-540-45247-$8_1$2.

[16] H. O. Kerkar, M. N. Doja, "Using deep convolutional neural networks for image steganalysis," IEEE Access, vol. 8, pp. 47418-47434, 2020, doi: 10.1109/ACCESS.2020.2974456.

[17] X. Zhao, A. T. S. Ho, "Machine Learning for Image Steganalysis: A Review," Journal of Image and Vision Computing, vol. 96, pp. 103912, March 2020, doi: 10.1016/j.imavis.2020.103912.

[18] F. Balado, "Information Hiding: Challenges for Forensic Experts," Computers Security, vol. 31, no. 3, pp. 418-435, April 2012, doi: 10.1016/j.cose.2011.12.004.

[19] J. Fridrich, R. Du, "Secure steganographic methods for palette images," in Proceedings of the 3rd Information Hiding Workshop, pp. 47-60, Sep. 1999, doi: 10.1007/3-540-46786-7_4.

[20] L. Liu, H. Tan, "A Survey of Image Steganography and Steganalysis Techniques in Multimedia," Multimedia Tools and Applications, vol. 77, no. 15, pp. 20427-20463, August 2018, doi: 10.1007/s11042-018-5731-4.

[21] T. Pevný, P. Bas, J. Fridrich, "Steganalysis by Subtractive Pixel Adjacency Matrix," IEEE Transactions on Information Forensics and Security, vol. 5, no. 2, pp. 215-224, June 2010, doi: 10.1109/TIFS.2010.2045849.

[22] B. Li, M. Wang, J. Huang, X. Li, "A new cost function for spatial image steganography," in IEEE International Conference on Image Processing (ICIP), pp. 4206-4210, Oct. 2014, doi: 10.1109/ICIP.2014.7025858.

[23] A. D. Ker, "Steganalysis of LSB Matching in Grayscale Images," IEEE Signal Processing Letters, vol. 12, no. 6, pp. 441-444, June 2005, doi: 10.1109/LSP.2005.849493.

[24] Y. Zhang, C. Qin, F. Liu, X. Luo, "Deep Learning for Steganography and Steganalysis: A Survey," WIREs Data Mining and Knowledge Discovery, vol. 10, no. 3, e1348, 2020, doi: 10.1002/widm.1348.

[25] S. Tan, B. Li, "Matrix Embedding for Large Payloads," IEEE Transactions on Information Forensics and Security, vol. 12, no. 1, pp. 170-183, Jan. 2017, doi: 10.1109/TIFS.2016.2603961.

[26] H. Xu, W. Zhang, "Deep Convolutional Neural Network to Detect J-UNIWARD," in Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, pp. 67-72, June 2017, doi: 10.1145/3082031.3083240.

[27] O. Taran, S. Bonev, T. Holotyak, T. Pun, "Image Steganography and Deep Learning: A Survey of Challenges, Techniques and Opportunities," Frontiers in Computer Science, vol. 2, Article 7, Feb. 2020, doi: 10.3389/fcomp.2020.00007.

[28] R. Cogranne, C. Zitzmann, L. Fillatre, I. Nikiforov, P. Cornu, "Reliable Detection of LSB Steganography in Color and Grayscale Images," in Proceedings of the ACM Workshop on Multimedia and Security, pp. 97-106, Sep. 2011, doi: 10.1145/2072298.2072314.

[29] V. Holub, J. Fridrich, "Designing Steganographic Distortion Using Directional Filters," in IEEE Workshop on Information Forensics and Security (WIFS), pp. 234-239, Nov. 2012, doi: 10.1109/WIFS.2012.6412637.

[30] N. Johnson, S. Jajodia, "Exploring steganography: Seeing the unseen," IEEE Computer, vol. 31, no. 2, pp. 26-34, Feb. 1998, doi: 10.1109/2.660187.