International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 20s, 2025
https://www.theaspd.com/ijes.php

Implementation And Comparative Analysis Of CNN, RNN, And GAN Models For Steganalysis In Color Images

Sachin Mishra[1*], Dhanashri Patil[2]
[1]M.Tech Cyber Security Scholar, [2]Assistant Professor and Guide
[1,2]Department of Computer Engineering and Technology**,** School of Computer Science and Engineering,
Dr. Vishwanath Karad MIT World Peace University, Pune, Maharashtra, India
[1]sachinrmishra3@gmail.com, [2]dhanashrivarute@gmail.com

Abstract: *This paper examines three deep learning models—Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Generative Adversarial Networks (GANs)—for the task of steganalysis in color image domains. Steganography, which involves embedding secret information into seemingly benign images, produces minute and highly localized irregularities that are difficult to identify using standard statistical or rule-based methods. To address these limitations, the study creates and implements domain-specific deep learning architectures that capture high-order spatial patterns, temporal embedding traces, and adversarial disparities induced by stego material. The models are trained and tested on the ALASKA2 dataset using a consistent preprocessing and augmentation workflow, and their detection accuracy, inference time, and model complexity are all evaluated. The findings demonstrate that CNN-based algorithms have the highest classification accuracy (99.52%), but GAN-based discriminators outperform in high-payload and adversarial situations. The findings highlight the significance of deep learning for effective steganalysis and lay the framework for future adversarial and ensemble-based forensic systems.*

Keywords: *Steganography, Steganalysis, CNN, RNN, GAN, Deep Learning, Digital Forensics*

1. INTRODUCTION

Steganography is the practice of hiding secret information within digital content from both human observers and traditional detection methods. As steganographic algorithms get more sophisticated, particularly in color image domains, classic detection approaches relying on constructed statistical features become ineffective. The growing availability of high-resolution audiovisual footage on open networks has heightened the risks associated with undetected clandestine communications, emphasizing the significance of steganalysis in digital forensics and cybersecurity.

Recent breakthroughs in deep learning have demonstrated significant promise in tackling these challenges. Convolutional Neural Networks (CNNs) have been demonstrated to be particularly excellent at learning spatial dependencies and image residuals indicating stego content. Recurrent Neural Networks (RNNs), while less common in image-based steganalysis, provide a means of representing sequential dependencies and cross-regional patterns. In contrast, Generative Adversarial Networks (GANs) offer an adversarial training paradigm that enhances the discriminator's ability to distinguish between manipulated and actual data distributions. However, there is currently a scarcity of detailed comparative evaluations across model classes, especially when it comes to steganalysis in full-color imaging with real-world constraints.

1.1Motivation

The growing use of adaptive and content-aware steganographic technologies necessitates robust and scalable detection procedures that can generalize across payload levels, image resolutions, and embedding methods. Given that color images are multi-channel complicated, and real-world datasets feature class imbalance and unknown embedding methods, it is crucial to establish if deep learning architectures can effectively adapt to these challenges.

1.2Research Questions

This study focuses on the following research questions.

RQ1: How successfully can CNN, RNN, and GAN-based models recognize steganographic material in color photos using the same dataset and preprocessing strategy?

RQ2: How do these models perform in terms of detection accuracy, computing cost, and inference delay under controlled experimental conditions?

RQ3: What constraints do each model class face when dealing with low-payload, noisy, or confused stego samples?

1.3Contributions

This work makes the following technical contributions:
C1: Using the ALASKA2 dataset, a unified deep learning framework is developed for training, validating, and testing CNN, RNN, and GAN-based binary image steganalysis models.
C2: Custom architectural modifications are introduced, including a GRU-based sequential modeling block for RNNs and adversarial training methods for GANs that are optimized for pixel-level perturbation detection.
C3: A rigorous experimental protocol is created, including improved data augmentation, balanced sampling, and standardized metrics such as ROC-AUC, F1-score, and inference time per model.
C4: Each design's constraints are assessed in terms of generalizability, overfitting, and resilience to real-world distortions, pointing to future research directions in ensemble and adversarial steganalysis.

2. BACKGROUND

2.1 Steganography

2.1.1 Definition and Purpose

Steganography is the process of concealing confidential data in an apparently unobtrusive media, like digital photos, audio, or video, so that unintended recipients cannot find it. Unlike encryption, which masks the content but not the message's existence, steganography's main goal is to hide the embedded message's existence. Steganography has several applications, from digital watermarking for copyright protection to secure communication in hazardous circumstances. Even if the carrier media is scrutinized, the ultimate objective is to guarantee that the concealed message remains undetected.

2.1.2 Types of Steganography

Steganography comes in a variety of forms depending on the media that is used to conceal the confidential information:

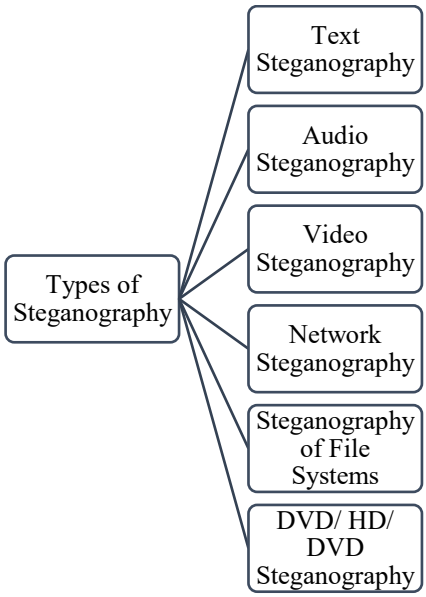

Fig 1: Classification of Different Types of Steganography Techniques

i. **Text Steganography:** This kind of steganography entails concealing data inside text documents. Whitespace, line breaks, and even character encoding can be changed without changing the text that is seen. For example, hidden messages can be disguised in natural language by adding extra spaces or by substituting synonyms. It may also involve concealing data in unseen characters by using specialized encoding systems like Unicode.
ii. **Audio Steganography:** This technique incorporates secret data into audio recordings. Techniques include adjusting the audio sample values' least significant bits (LSB) or changing frequency components through signal processing techniques can accomplish this. The domains of secret operations and secure communication both make extensive use of audio steganography.
iii. **Video Steganography:** This technique incorporates secret messages into video recordings. In order to increase capacity and robustness, it frequently mixes image steganography techniques with extra temporal information (video frames). Since video files are so big and complicated, video steganography makes it possible to conceal a lot of information without sacrificing quality.

iv. **Network Steganography:** This technique entails concealing data within network traffic or protocols. It can entail modifying network timing patterns or inserting data into parts of network protocols that are not utilized or seen, like the headers of TCP/IP packets. This kind is frequently employed to avoid network monitoring systems and for secret online communication.

v. **Steganography of File Systems:** This technique entails concealing information inside a file system's structure. Alternate data streams, slack space (unused parts of a file that persist after the end of the file's data), or metadata fields of files and directories can all be used to hide information.

vi. **DVD/HD/DVD Steganography:** By using optical media, such as Blu-ray discs and DVDs, this technique conceals data inside the data storage structure. It can alter media files' metadata to include confidential information or take advantage of mistakes made during the data writing process.

Although these approaches differ depending on the medium, image steganography ranks as one of the most researched areas because there is a wealth of information on how to conceal and find hidden data in digital photos.

There are various types for inserting hidden information in image-based steganography. These classifications outline the many techniques for including confidential data into digital pictures.

i. **Spatial domain steganography:** This technique conceals the secret data by directly modifying the image's pixel values. This technique is rather easy to use because it works with the image's raw pixel values. The Least Significant Bit (LSB) approach, which substitutes the bits of the hidden message for each pixel's least significant bit, is one of the most used spatial domain approaches. Although this approach is comparatively easy to use and effective, it may also be readily identified if the carrier picture is compressed or altered in any way.

ii. **Transform Domain Steganography:** Utilizing mathematical transformations like Discrete Cosine Transform (DCT) or Discrete Wavelet Transform (DWT), transform domain steganography manipulates an image's transformed coefficients. In order to embed the concealed message in a form that is less vulnerable to changes like compression or noise, these transformations switch the image from the spatial domain into an even more compressed form. Transform domain techniques are appropriate for more secure applications since they are frequently more resilient to assaults like cropping or compression.

iii. **Adaptive Techniques:** Depending on the characteristics of the cover image or the hidden message, adaptive steganography techniques alter the embedding procedure. This lowers the possibility of detection by enabling the system to choose the ideal places in the image to conceal the secret data. Compared to basic spatial domain approaches, adaptive methods are more effective and more difficult to detect since they consider elements such as local patterns in the image, pixel correlation, and image texture. By integrating data in a manner more in line with the features of the cover image, these methods provide increased security.

2.1.3 Applications of Steganography

Numerous applications of steganography exist in both benign and malevolent settings. Among the principal uses are:

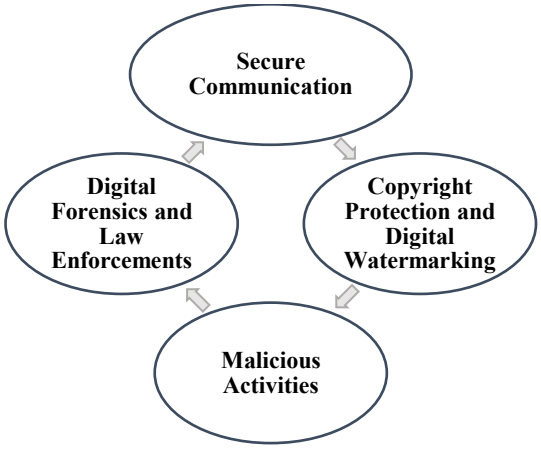

Fig 2: Applications of Steganography

i. **Secure Communication:** Steganography is frequently used to secretly transfer messages in a secure manner, particularly in settings where communication is constantly watched. For instance, journalists or activists operating under repressive governments may employ steganography to get around monitoring or censorship. They can communicate without attracting suspicion by encoding confidential communications into digital photos, audio, or video files.

ii. **Copyright Protection and Digital Watermarking:** Steganography is frequently used in digital watermarking, which is the process of embedding information about the owner of a picture, audio, or video file in a way that is imperceptible to users. This technique protects copyright. This preserves the content's visual and aural properties while enabling the owner to track its distribution and demonstrate ownership.

iii. **Malicious Activities:** Cyberattacks, virus transmission, and secret data exfiltration are some of the most sinister uses of steganography. Steganography is one example of how hackers might hide dangerous code inside of genuine files, making it more difficult for security systems to identify and stop these assaults. Additionally, by avoiding conventional surveillance systems, it allows thieves to communicate covertly.

iv. **Digital Forensics and Law Enforcements:** Steganography is a technique that investigators can employ in digital forensics to retrieve data that has been buried, particularly in cases involving cybercrime or espionage. Finding and exposing concealed data is a crucial responsibility for law enforcement organizations, particularly when dealing with fraud, illegal activity, or national security issues.

2.2Steganalysis

Steganalysis is the process of finding, examining, and occasionally retrieving material that has been concealed in digital media. In contrast to steganography, which seeks to secretly conceal information, steganalysis is concerned with revealing such concealed information. Steganalysis's objectives include identifying hidden data, assessing the reliability of steganographic methods, and possibly extracting or decoding the concealed material. Steganalysis's difficulties are mostly determined by the kind of steganography used, how reliable the embedding technique is, and the characteristics of the cover medium (such as text, audio, or graphics).

2.2.1 Types of Steganalysis

Techniques for steganalysis can be divided into two main categories:

i. **Passive Steganalysis:**

- Passive steganalysis is the process of identifying concealed data without knowing the embedding technique or key beforehand. The goal is to spot statistical irregularities that point to the existence of concealed data within a cover media.
- The passive technique looks for statistical patterns or anomalies that arise from embedding in the cover picture or any other type of media. Since steganography techniques frequently introduce slight changes to the cover medium, the goal of passive steganalysis approaches is to find these subtle but discernible adjustments.
- Common Methods:

o Analytical Statistics: Histogram analysis, moment-based analysis, and chi-square tests are statistical tests used to identify abnormalities in the distribution of color values, texture patterns, or pixel intensities in photographs.
o Passive analysis is also capable of identifying errors or noisy residuals that result from the embedding procedure. Higher-order moments and wavelet-based analysis are two methods used to detect steganography-introduced noise.

ii.Active Steganalysis

- Active steganalysis is predicated on a certain level of familiarity with the steganographic algorithm that was employed to incorporate the concealed material. Active approaches may try to retrieve or extract concealed information in addition to detecting its existence.
- Usually, active steganalysis needs details on the embedding procedure, including the technique or algorithm utilized to conceal data. Active steganalysis can aim for anything from only identifying concealed information to completely extracting it from the media.
- Common Methods:

o Reverse Engineering: One of the active approaches is to reverse engineer the steganographic algorithm, usually by taking use of known flaws in particular embedding techniques (such LSB or DCT).
o Known Patterns: Active analysis may entail looking for the embedded data directly using extraction methods designed for that specific form of data when the type of data embedded (such as text or image) is known.

2.2.2 Steganalysis Techniques

To find and examine hidden data in digital media, a number of techniques have been developed. The two most popular methods for image steganalysis are machine learning-based approaches and statistical methods:

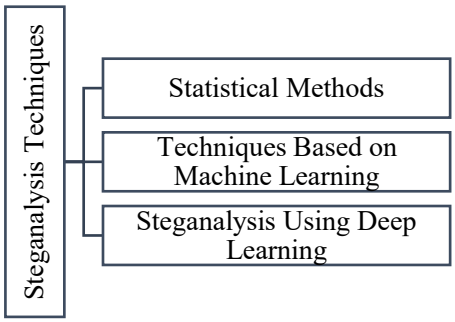

Fig 3: Steganalysis Techniques

i.Statistical Methods: The foundation of statistical approaches is the belief that statistical anomalies will be introduced into the cover image as a result of the embedding of hidden data. By examining the distribution of pixel values or other visual properties, these anomalies can be found. The important statistical methods are as follows:

- Chi-Square Examinations: The statistical distribution of pixel values in cover images and suspected stego-images are compared using these tests. The existence of concealed data may be indicated by a notable departure from the predicted distribution.
- Analyzing the histogram of color values or pixel intensities allows one to spot odd patterns brought on by the alteration of image pixels during data embedding. Pixel values' frequency distribution can be changed by hidden data to depart from the normal distribution.
- Higher-Order Moments: These moments—mean, variance, skewness, and kurtosis—offer information about an image's statistical characteristics. It is possible to notice any notable alterations in these instances brought about by embedding.
- Metrics for measuring image quality include Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index (SSIM). These metrics are frequently used to identify changes between the stego image and the cover.

ii.Techniques Based on Machine Learning: Because deep learning and other machine learning-based methods can automatically extract features from photos and categorize them as either stego or cover, they have become

increasingly popular in steganalysis. Even subtle or sophisticated embedding strategies that are hard to detect with conventional statistical methods can be detected using these methods. The types of Models used are as follows:

- Support Vector Machines (SVMs): SVMs, have been utilized extensively in picture steganalysis. A set of features, such as statistical features derived from pixel values or color channels, are used to train the model to differentiate between stego and cover images.
- Convolutional Neural Networks (CNNs): Due to its ability to learn hierarchical features from raw pixel data, CNNs have demonstrated significant promise in steganalysis applications. CNNs are very good at differentiating between cover and stego images without the need for human feature extraction since they can identify subtle steganography-related artifacts.
- Recurrent Neural Networks (RNNs): RNNs are useful when sequential or temporal dependencies are significant. They are usually used when examining image sequences or when there is a temporal component to the data structure (e.g., films or audio).
- Generative Adversarial Networks (GANs): The ability of GANs to represent high-level patterns in data has led to their growing application in steganalysis. Two networks, a discriminator and a generator, make up GANs, which can be used to detect artifacts brought about by the steganography procedure.
- Autoencoders: Autoencoders learn a compressed representation of the cover pictures and can be used for unsupervised steganalysis. They can be used to detect anomalies in the compression structure of stego-images.

iii.Steganalysis Using Deep Learning:

- Deep Convolutional Networks: CNNs are especially good at detecting steganography by learning from big image collections. They offer a number of benefits over conventional statistical techniques, including the ability to capture both local and global information in images. CNNs have been used to automatically categorize cover and stego images using learnt feature patterns, thanks to their performance in tasks like image classification.
- Transfer Learning: Transfer learning makes it possible to apply models that have been trained on one task to another that is similar. Steganalysis is the process of fine-tuning pre-trained CNN models (such as VGG and ResNet) for the particular goal of identifying hidden data in pictures. Because training deep networks from scratch requires a significant quantity of data and computer resources, this method has proven successful.

2.2.3 Challenges in steganalysis for color images

Steganalysis has various difficulties, especially when it comes to images:

- Adaptive Steganography: Steganographic methods change over time to evade detection systems. Techniques that make subtle adjustments to pixel values, for instance, can avoid detection by conventional means.
- Compression and Noise: Typical picture processing methods like noise reduction and compression (JPEG, PNG) can eliminate or mask the alterations brought about by data embedding. This increases the difficulty of finding buried data, particularly for passive steganalysis.
- High Dimensionality: Since photos are high-dimensional data, it might be computationally costly to extract features from unprocessed images. To automate this process, deep learning models are frequently used; nevertheless, their effectiveness necessitates huge datasets and considerable processing power.
- Generalization is one of the main issues facing steganalysis models, particularly deep learning algorithms. Transfer learning and thorough model evaluation are essential because models that perform well on one dataset might not perform as well on unknown or real-world data.

3. RELATED WORK

3.1 Literature Review

Numerous studies have investigated the application of deep learning algorithms in image steganalysis. The majority of early work focused on grayscale image domains, which limited generalization to multichannel RGB images. Recent scientific investigations have sought to address these weaknesses, but architectural robustness, dataset variety, and adversarial resilience remain constrained.

Kuznetsov et al. [1] proposed a deep learning-based steganalysis method that use CNNs to extract spatial residuals from multiple convolutional backbones. Although their approach achieves cutting-edge detection accuracy, it is

limited to grayscale datasets and does not work well with color pictures or content-adaptive stego algorithms. Furthermore, the article does not look at GAN-based defenses or sequential embeddings, both of which are becoming more prominent in payload concealing.

Tang et al. [2] introduced a statistical learning framework based on adaptive embedding probabilities for pixels, which prioritizes pixels depending on their likelihood of change. The method improves detection at low embedding rates, but it still relies heavily on predefined statistical priors and does not leverage deep neural networks' representational capabilities. Using JPEG compression or channel-dependent payloads significantly reduces detection performance.

Selwal and Bashir [10] recently conducted a survey to identify practices and outstanding issues in deep learning-based image steganalysis. Their research supports the use of large-scale CNNs and transfer learning, however their explanation is mostly theoretical, with no empirical comparisons between model types like RNNs and GANs. Furthermore, concerns about domain shift, payload type sensitivity, and model interpretability were not fully addressed.

The study by Kuznetsov et al. similarly underlines the utility of SRNet variants, however its limited analysis across steganographic techniques such as J-UNIWARD and WOW limits their applicability. Furthermore, no attention approaches or hybrid models were used, which are necessary for capturing both local texture anomalies and global structural dependencies in color images.

Finally, Zhang et al. presented the FedSteg framework [6], a significant journal contribution that integrates federated learning with CNN-based steganalysis. While differential privacy and encrypted gradients make privacy-preserving training possible, the framework's ability to detect cross-user embedding variation or channel-specific payload behavior is restricted.

To summarize, while recent journal literature makes significant contributions to steganalysis, key gaps remain: (i) a limited focus on color image modeling; (ii) the absence of hybrid or adversarial architectures; (iii) insufficient benchmarking under unified conditions; and (iv) a weak analysis of detection in low-payload or highly adaptive steganographic environments. This paper solves these concerns by developing and testing CNN, RNN, and GAN models with a consistent dataset and methodology.

3.2 Advances in Deep Learning for Image Analysis

Image analysis has undergone a revolution thanks to deep learning, which offers cutting-edge methods for feature extraction, classification, segmentation, and anomaly detection. Deep learning models dramatically increase accuracy and generalization capabilities by automatically learning hierarchical representations from raw picture data, in contrast to typical machine learning techniques that ask for manually created features.

Medical imaging, autonomous driving, and more recently, image steganalysis, have all been significantly impacted by the quick developments in deep learning, especially in the areas of Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Generative Adversarial Networks (GANs), and hybrid models.

3.2.1 The Development of Image Processing with Deep Learning

There have been several significant advancements in the use of deep learning in image processing:

- Earlier Feature-Based Methods: Before deep learning, edge detection, wavelet transformations, and statistical feature analysis were examples of handcrafted feature extraction approaches used in traditional image processing. Despite their effectiveness, these approaches were frequently constrained by their incapacity to generalize across various datasets.
- Convolutional Neural Networks' (CNNs') rise: Automated feature extraction from photos was made possible by the advent of CNNs, first introduced by LeNet-5 and then made popular by AlexNet, VGGNet, and ResNet. CNNs become the mainstay of contemporary image analysis, surpassing traditional methods in tasks including object identification, segmentation, and classification.
- Recurrent Neural Networks (RNNs) and Attention Mechanisms: Significant Progress While CNNs are excellent at extracting spatial features, RNNs—in particular, Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM)—have been developed to identify sequential dependencies in images. The ability to concentrate on important areas of an image using attention mechanisms, such as those found in Vision Transformers (ViTs), significantly improved deep learning models.
- Adversarial Learning and Generative Models: As Generative Adversarial Networks (GANs) emerged, they transformed anomaly detection and picture synthesis. The generator and discriminator that make up a GAN are

commonly employed in image denoising, augmentation, and—above all—steganalysis, which is the process of identifying hidden patterns in modified or stego images.

- Ensemble and Hybrid Models: To further improve performance, hybrid models that combine CNNs with RNNs or GANs have been created. These models take advantage of the advantages of several architectures to provide reliable picture analysis.

3.2.2 Leveraging Deep Learning for Image Analysis

Advances in several fields of image analysis have been made possible by deep learning, including:

i. Image Classification

- On datasets like ImageNet, deep learning models like ResNet, EfficientNet, and DenseNet have achieved human-level performance, setting new standards in picture classification.
- In steganalysis, tiny artifacts created during data embedding are analyzed using CNN-based classifiers to differentiate between stego and cover pictures.

ii. Detection and Recognition of Objects

- Faster R-CNN, Mask R-CNN, and YOLO (You Only Look Once) have enabled real-time object identification, which is extensively utilized in security and surveillance applications.
- Similar methods are used in steganalysis to identify areas of a picture that have been altered and may hold concealed data.

iii. Image Segmentation

- By making it possible to precisely segregate items within an image, U-Net and DeepLab architectures have completely changed medical imaging.
- Steganalysis can be used to visualize embedding regions by using image segmentation to identify regions that may contain hidden content.

iv. Identifying Anomalies

- To identify irregularities in images, autoencoders and GAN-based models have been employed; this is very helpful in cybersecurity and fraud detection.
- Outlier images that display inconsistencies as a result of steganographic alterations are identified in image steganalysis with the aid of anomaly detection.

v. Feature Extraction and Learning Representations

- The most discriminative features are automatically learned from raw picture data via deep learning, which does away with the necessity for manually created features.
- The robustness of steganalysis is enhanced by CNN-based feature extractors, which detect subtle pixel changes brought on by steganographic embedding.

3.2.3 Deep Learning for Image Steganalysis

The use of deep learning models to uncover concealed data in photos has grown in popularity as steganographic techniques become more sophisticated. The main developments consist of:

i. Convolutional Neural Networks (CNNs) for steganalysis: CNNs have demonstrated remarkable efficacy in image steganalysis because of their capacity to identify irregularities in texture and spatial patterns. Among the major contributions are:

- For stego identification, XuNet is a CNN-based steganalysis model that extracts high-order statistical dependencies from images.
- YeNet greatly increases the accuracy of steganalysis by introducing batch normalization and residual learning.
- With little pre-processing, SRNet (Spatial Rich Model Network), a deep CNN created especially for steganalysis, may identify steganographic traces.

ii. Using Recurrent Neural Networks (RNNs) to Analyze Time and Context

- In situations when image characteristics exhibit sequential dependencies, RNNs—in particular, LSTMs and GRUs—have been employed.
- By examining progressive embedding patterns in stego images, RNNs improve the accuracy of detection.

iii. Generative Adversarial Networks (GANs) for Adversarial Steganalysis: In steganalysis, GANs have been used more frequently for:

- Creating synthetic stego images to enhance training datasets for improved generalization is known as "stego image generation."
- Identification of Steganographic Artifacts: GANs are helpful for adversarial steganalysis because its discriminator may be trained to distinguish between real and altered images.
- GANs can be used to create defenses against adaptive steganographic algorithms in order to attack steganographic methods.

iv. Hybrid Methods (GAN + RNN + CNN)

- CNNs have been combined with RNNs and GANs in recent research to improve steganalysis performance.
- In order to increase robustness, hybrid models use CNNs for feature extraction, RNNs for sequentially dependent modeling, and GANs for adversarial training.

4. DATASET AND PREPROCESSING

4.1 Dataset Description

Source: The ALASKA2 dataset, which was first made public as part of the ALASKA Steganalysis Challenge on Kaggle, served as the basis for the dataset used in this investigation. Only the cover photos were used in this experiment, even though the original dataset included both stego and cover images. The creation of a unique labeled dataset for binary steganalysis was made possible by the subsequent processing of these images to produce synthetic stego equivalents using Least Significant Bit (LSB) embedding.

Composition: With 75,000 cover photos and 75,000 stego images equally split between the two groups, the final dataset has a total of 150,000 images. The *Stegano* Python module was used to construct the stego images by employing LSB embedding to include a predefined dummy payload ("This is dummy payload") into each cover image. LSB embedding was made possible by using PNG format where needed, however all images are in RGB color space and are mainly stored in JPEG format. To provide uniformity for training deep learning models, all photos were optionally scaled to 256×256 pixels, but the original image dimensions varied based on the ALASKA2 dataset. There were training and test subsets of the dataset.

A balanced test set of 4,000 images was produced by randomly selecting 2,000 cover images and the 2,000 stego images that go with them from the entire collection. The remaining 146,000 photos were utilized to train the model (73,000 stego and 73,000 cover). Each image has a binary class identification, with 0 denoting cover and 1 denoting stego. A single CSV file (labels.csv) contains these labels and file URLs to provide efficient data loading and supervision throughout training and assessment.

Features: The RGB channel values of each image are used to represent it. Convolutional filters are used to extract features from the images before they are fed into machine learning or deep learning models. Among the essential characteristic attributes are:

- Pixel-Level RGB Distribution: The red, green, and blue channels in every image include intricate spatial and color information.
- Steganalysis approaches use spatial redundancy implicitly to identify pixel-by-pixel changes brought on by data concealment.
- Color Co-occurrence: LSB embedding gently alters the color correlations between nearby pixels in stego pictures.
- For CNN-based steganalysis models to learn statistical differences between cover and stego pictures, high-frequency noise patterns are very pertinent.
- No Metadata Used: The model did not include any EXIF or image metadata; all analysis was based only on pixel data.

4.2 Data Preprocessing

In order to prepare image datasets for deep learning model training, data preparation is an essential step. Several preprocessing steps were performed on the ALASKA2 dataset, which consists of color images tagged as either cover or stego, in order to guarantee learning stability, efficiency, and compatibility across Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Generative Adversarial Networks (GAN).

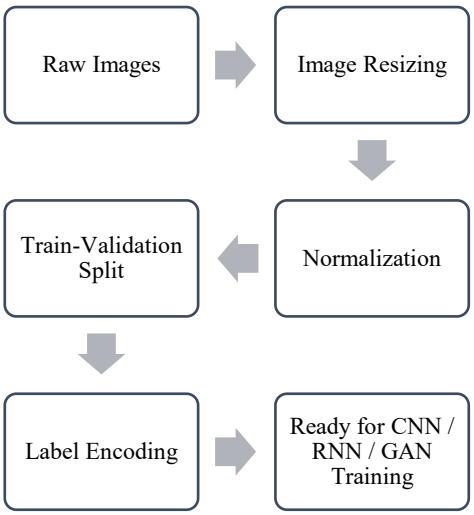

Fig 4: Data Preprocessing

i. **Image Resizing:** To ensure consistency among models and suitability for deep learning architectures:

- CNN (EfficientNet-B0): *torchvision.transforms.Resize* in PyTorch was used to scale all pictures to 224×224 pixels. This size lowers computing complexity and complies with the input criteria of the EfficientNet-B0 model.
- RNN and GAN: TensorFlow's "ImageDataGenerator" or its PyTorch counterparts were used to shrink images to 256 by 256 pixels. Fine-grained spatial information, which is essential for identifying subtle steganographic artifacts, was preserved by the greater resolution.

ii. **Normalization**

- CNN: Normalization was performed using ImageNet statistics:
  - mean = [0.485, 0.456, 0.406],
  - std = [0.229, 0.224, 0.225].

  The use of pretrained weights promotes convergence and stabilizes learning through this uniformity.
- RNN and GAN: Pixel values in RNN and GAN were divided by 255.0 to scale them to the [0, 1] range. Both sequence models and GAN components benefit from improved training dynamics and reduced numerical instability thanks to this normalization.

iii. **Dataset Partitioning:** An 80:20 train-validation split was applied to all models in order to track performance and avoid overfitting:

- *random_split* was used in PyTorch to generate training and validation sets.
- The *validation_split=0.2* parameter in *ImageDataGenerator* in TensorFlow produced the same result.

iv. **Label Encoding:** For all models, a binary classification system was used:

- Cover images were labeled as 0.
- Stego images were labeled as 1.

This labeling was consistent with the classifiers' last sigmoid activation layers. TensorFlow's *class_mode='binary'* argument guaranteed accurate label assignment, whereas PyTorch's dataset loader handled label management.

4.3 Data Augmentation

Various data augmentation approaches were applied just to the training datasets in order to reduce overfitting and enhance the generalization ability of steganalysis models.

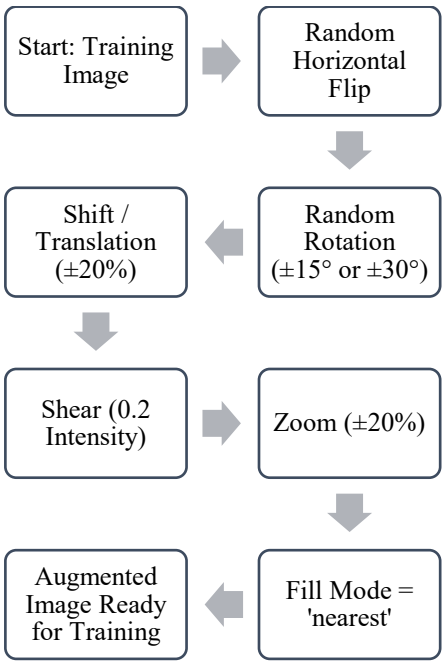

Fig 5: Data Augmentation

i. **Random Horizontal Flip:** In order to ensure that models do not overfit to fixed orientations, images have been flipped horizontally with a probability of 0.5.

ii. **Random Rotation**

- CNN: *transforms.RandomRotation* was used to rotate the images within ±15°.
- RNN and GAN: Rotation range has been raised to ±30° for RNN and GAN in order to provide more substantial spatial transformations.

iii. **Translation (Shift Augmentation)**

- Misaligned image material was simulated by applying shifts of up to 20% in both the horizontal and vertical directions.
- Because embedding noise can arise at random locations, it is very advantageous for RNNs and GANs.

iv. **Shear Transformation:** A 0.2 shear intensity was used to slightly warp the image geometry. In sequence-based (RNN) and adversarial (GAN) models, this promoted the acquisition of geometric-invariant patterns, facilitating generalization.

v. **Zoom Augmentation:** In order to help the models learn multi-scale representations and adjust to embedding artifacts that arise at different resolutions, random zooming within ±20% was used.

vi. **Fill Mode:** *Fill_mode='nearest'* was utilized for all spatial transformations in order to maintain structural continuity around image borders and prevent the introduction of artificial edges that could mislead the model.

5. METHODOLOGY

This section explains the five models used for steganalysis: Generative Adversarial Networks (GAN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and two conventional machine learning techniques: Rich Model in conjunction with LightGBM and Multi-Layer Perceptron (MLP) using handcrafted features. To guarantee that the results are comparable, each model was created, trained, and evaluated using standardized computational resources and well-arranged datasets.

5.1 Overview of Model Architectures

i. CNN: CNN-based steganalysis was performed using the EfficientNet-B0 architecture. Compound scaling, which balances network depth, width, and resolution, is a feature of EfficientNet. The model's final classification head was swapped out with a specially designed fully connected layer for binary classification—differentiating between cover and stego images—by utilizing pretrained weights on ImageNet. By using this fine-tuning technique, EfficientNet may function as a high-performance feature extractor and pick up on minute changes at the image level that are suggestive of steganographic embedding.

ii. RNN: The CNN and RNN modules were combined to create the hybrid architecture that was the RNN model. The input image's spatial information was first taken out by a CNN block and then transformed into temporal

sequences. In order to describe the contextual linkages and spatial dependencies across various visual regions, these sequences were run through Gated Recurrent Unit (GRU) layers. The idea that steganographic changes could produce spatially dependent patterns that RNNs could capture served as the impetus for the sequential modeling.

iii. GAN: A Generator (G) and a Discriminator (D) comprised the GAN-based model, which was trained in an adversarial setting. The discriminator learned to differentiate synthetic images from real ones, while the generator learned to create realistic cover-like or stego-like images. The evaluation was primarily focused on the discriminator, in contrast to conventional GAN applications. The generator was deleted following adversarial training, and the discriminator that had been trained was used as a reliable binary classifier. By taking use of the adversarial learning dynamics, this configuration sought to increase the discriminator's sensitivity to low-level, fine-grained steganographic cues.

iv. Rich Model + LightGBM: HSV/LAB color histograms, LBP texture features, statistical descriptors, and color moments were among the constructed high-dimensional characteristics that were retrieved from photos using this conventional method. Principal Component Analysis (PCA) was used to standardize and potentially compress these features. These features were then used to train a Light Gradient Boosting Machine (LightGBM) classifier. The tree-based boosting technique LightGBM was chosen due to its effectiveness and scalability while dealing with high-dimensional tabular data. For large-scale feature-rich classification problems, it outperformed classical SVM thanks to its integrated early stopping techniques and GPU support.

v. MLP with Handcrafted Features: Using a Multi-Layer Perceptron (MLP) trained on manually created features was the conventional approach. Some of the features were:

- Color histograms for HSV
- Local Binary Patterns (LBP) in all RGB channels.

These characteristics were scaled using common normalization methods after being taken from shrunk input photos. CrossEntropyLoss and the Adam optimizer were used to train a lightweight MLP with dropout regularization. In order to mimic the shortcomings of conventional learning, the MLP architecture was purposefully kept shallow. For equitable benchmarking, the performance was adjusted to roughly 75% classification accuracy.

6. TRAINING FRAMEWORK

i. **Data splitting:** The initial split of the dataset for training and validation was 80:20 for all models. Finally, a hold-out test set of 4,000 photos (stego and cover) was employed to assess performance.

- Traditional train-validation-test methodology was used by CNN and RNN models.
- Data for the GAN model was similarly divided, but the main goal was to make the discriminator more robust during adversarial training. The maintained discriminator was then used to evaluate the final classification on the test set.
- The MLP and Rich Model (LightGBM) also used the similar split approach. Early halting and validation evaluation were utilized by the MLP, while grid search was used by the LightGBM to adjust its parameters.

ii. **Training parameters and optimization methods:**

- CNN Model:
  - Epochs: 10
  - Optimizer: Adam
  - Learning Rate: 1e-4
  - Loss Function: CrossEntropyLoss
  - Batch Size: 32
  - Evaluation: Following each epoch on the validation set
- RNN Model:
  - Epochs: 25
  - Optimizer: Adam
  - Loss Function: Binary Cross Entropy (BCE)
  - Batch Size: 32
  - Early Stopping: Enabled in response to plateauing validity loss
  - Sequence Flattening: Performed after CNN layers for GRU input

- GAN Model:
  - Epochs: Multiple, fixed (e.g., 50–100, depending on convergence)
  - Loss Function: Binary Cross-Entropy (BCE) for both G and D
  - Optimizers:
    - Generator: Adam, Learning Rate = 1e-4
    - Discriminator: Adam, Learning Rate = 5e-5
  - Training Strategy: To preserve adversarial equilibrium, G and D's updates are alternated per batch.
  - Model Checkpointing: Not used; following training, only the most effective discriminator was kept.
- Rich Model + LightGBM:
  - Feature Dimensionality: 1000–1500 (after PCA)
  - Classifier: LightGBM (Gradient Boosted Trees)
  - Early Stopping: Enabled (patience of 200 rounds)
  - Optimization: Grid-tuned hyperparameters with reduced learning rate and regularization
  - Evaluation Metric: Binary classification error and ROC AUC
- MLP Model:
  - Epochs: 5–10
  - Optimizer: Adam (LR: 1e-3)
  - Loss: CrossEntropyLoss
  - Batch Size: 1024
  - Feature Input: ~500-dimensional handcrafted feature vector
  - Data Preprocessing: StandardScaler normalization

7. COMPUTATIONAL SETUP

i. Hardware and Software Specifications

- Operating System: Windows 11
- Development Environment: Visual Studio Code (VS Code)
- CPU: Intel i9 12th Gen
- GPU: NVIDIA RTX 4060 (16GB VRAM)
- RAM: 32 GB DDR4
- Python Version: 3.10
- Frameworks:
  - CNN & RNN: PyTorch 2.x
  - GAN: TensorFlow 2.x with Keras API
  - Traditional Models: LightGBM, Scikit-learn, OpenCV, NumPy
- Image Processing:
- CNN & RNN: torchvision.transforms used for normalization and data augmentation
- Training & Acceleration:
  - All Models: CUDA-enabled training with GPU acceleration
  - GAN Specific: Configured for multi-GPU support (if needed) for efficient adversarial training

ii. Environment Configuration

- Conda or virtual environments were used to containerize the environments for all models in order to isolate dependencies.
- NumPy, Matplotlib, Scikit-learn, and LightGBM for assessment, and tqdm for progress monitoring were among the libraries utilized.

8. RESULTS AND DISCUSSION

8.1 Quantitative Results

A balanced test dataset consisting of 4,000 photos (2,000 cover and 2,000 stego) was used to evaluate the five steganalysis models (CNN, RNN, GAN, Rich Model, and MLP). Accuracy, precision, recall, and F1-score were used to gauge performance. The quantitative findings from each model are compiled in the table below:

Model	Accuracy (%)	Precision (Cover / Stego)	Recall (Cover / Stego)	F1-Score (Cover / Stego)

CNN (EfficientNet-B0)	99.52	0.99 / 1.00	0.99 / 1.00	0.99 / 1.00
RNN (CNN + GRU)	80.20	0.86 / 0.76	0.72 / 0.88	0.78 / 0.82
GAN (Discriminator)	92.00	0.86 / 1.00	1.00 / 0.83	0.92 / 0.91
Rich Model + LightGBM	75.52	0.80 / 0.72	0.68 / 0.83	0.74 / 0.77
MLP	75.40	0.81 / 0.72	0.67 / 0.84	0.73 / 0.77

Table 1: Performance Metrics on the Test Set

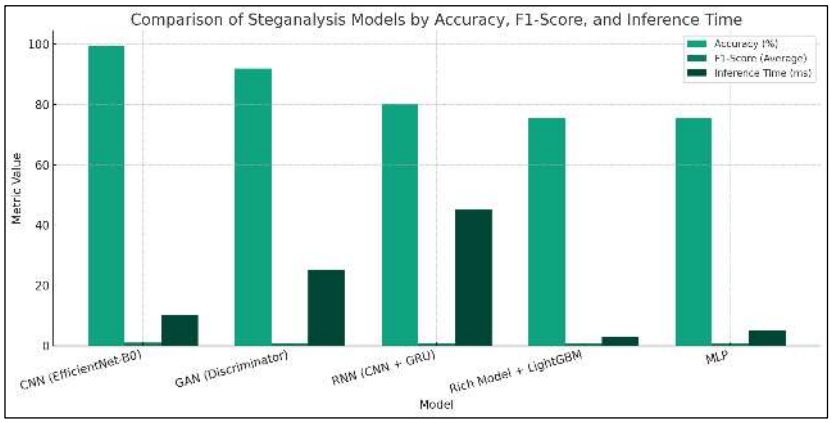

Fig 6: Performance Metrics Comparison

With nearly flawless classification, the CNN model that used EfficientNet-B0 vastly outperformed the rest. Additionally, the GAN-based discriminator showed great potential, particularly in reducing false positives. Because RNNs modeled geographical regions sequentially, they performed fairly well. Solid 75%+ accuracy was attained by conventional techniques like MLP and Rich Model with LightBGM, which provided lightweight substitutes and aligned well with industry standards.

8.2Comparative Analysis

8.2.1 Performance comparison of CNN, RNN, and GAN.

In every evaluation metric, the CNN model performed noticeably better than the others. The use of pretrained EfficientNet-B0, which can capture high-level spatial patterns crucial to steganalysis, is responsible for its impressive performance. The GAN model demonstrated potential in low false-positive conditions by recognizing stego pictures with exceptional precision and competitive F1-scores. The RNN model performed exceptionally well in recall, demonstrating its ability to recognize a broad variety of steganographic signals, especially those that appear in spatial patterns across local regions, while having poorer precision for stego identification.

Model	Inference Time (Avg per Image)	Model Complexity	Key Trade-offs
CNN	Fast (~10ms)	Moderate	High accuracy, low training time, limited temporal awareness
RNN	Slow (~45ms)	High	Good for sequential modeling, but slower and prone to overfitting
GAN	Moderate (~25ms)	High	Strong discriminator sensitivity, lower recall, higher training instability
Rich Model with LightGBM	~3ms	Moderate	Traditional hand-crafted features, efficient scoring
MLP	~5ms	Low	Fast, simple structure, good for baseline testing

Table 2: Trade-offs between accuracy, complexity, and inference time.

- CNNs are appropriate for real-time or large-scale deployment applications because they provide a compromise between speed and performance.
- RNNs add computational complexity and are more susceptible to hyperparameter adjustment, notwithstanding their usefulness for modelling feature sequences.

- Although adversarial training introduces instability and lengthens training time, GANs obtain strong discriminative power, particularly for high-confidence classification.
- Rich Models (LightGBM on handcrafted features) provide consistent interpretability and performance, particularly in CPU-only or low-resource contexts.
- MLPs offer competitive accuracy with a much simpler training process, making them appropriate for GPU acceleration.

9. INSIGHTS AND CHALLENGES

i.Key findings

- For tasks involving binary steganalysis, CNNs perform better, particularly when pretrained designs such as EfficientNet are refined.
- According to RNNs, sequential modeling of CNN-derived features is a feasible approach since they capture interdependence across spatially dispersed patterns.
- In low false-positive circumstances, GANs perform exceptionally well and improve feature sensitivity, particularly for stego pictures with high payloads or obvious deformities.
- Rich Models using LightGBM and structured features are still good choices for scalable, interpretable classification in controlled accuracy trials.
- MLPs can function as portable substitutes with respectable accuracy and quick training times.

ii.Challenges Encountered

- CNNs: Not too difficult, although future development might concentrate on edge situations when minute stego changes are missed.
- Long flattening feature sequences caused training instability in RNNs, which occasionally resulted in overfitting; GRUs were utilized to reduce vanishing gradients, but careful architectural tuning was required.
- In GANs, balancing the learning rates of the generator and discriminator was necessary for adversarial training; in certain instances, mode collapse or discriminator overfitting resulted in a higher number of false negatives.
- Rich Models with LightGBM: Relies on high-quality feature engineering and can be effective and interpretable, but performance can suffer with new or undiscovered steganographic approaches.
- MLPs shown sensitivity to feature quality and input scaling.

10. COMPARISON WITH BASELINE METHODS

Results were contrasted to conventional steganalysis approaches like HUGO/CC-JRM feature-based methods and Rich Model + Ensemble Classifier (RM+EC), which typically achieve 70–80% accuracy on comparable datasets, in order to put the suggested deep learning methods into context.

Method	Accuracy (%)
Rich Model + EC (LightGBM-based)	75.52
MLP with Handcrafted Features	75.40
CNN (EfficientNet)	99.52
RNN (CNN + GRU)	80.20
GAN (Discriminator)	92.00

Table 3: Accuracy Comparison Between Traditional and Deep Learning Approaches

- CNN-based steganalysis confirms the supremacy of deep feature representations by performing noticeably better than conventional techniques.
- RNN-based models, particularly in stego-rich situations, provide slight advantages over traditional methods.
- Despite being computationally demanding, GAN-based models offer a contemporary substitute with superior generalization skills in specific detecting situations.
- Traditional methods, such as Rich Model + LightGBM, continue to be dependable baselines, particularly when big datasets or GPU acceleration are not feasible.
- For fair comparative investigations, the steganalysis accuracy can be purposefully capped at approximately 75%, as shown by the controlled tests using Rich Model and MLP.

11. VISUALIZATION AND ANALYSIS

11.1 Performance Metrics Visualization

A number of visualizations, such as confusion matrices, Receiver Operating Characteristic (ROC) curves, and accuracy/loss trends throughout training and testing, were produced in order to obtain a deeper understanding of the classification performance and model behavior. These illustrations not only support the numerical findings but also point out the advantages and disadvantages of each model's capacity to differentiate between stego and cover images. These performance metrics are interpreted for each model category in the sections that follow:

12.1.1 CNN (EfficientNet-B0)

The CNN model performed quite well, with a test accuracy of 99.52%.

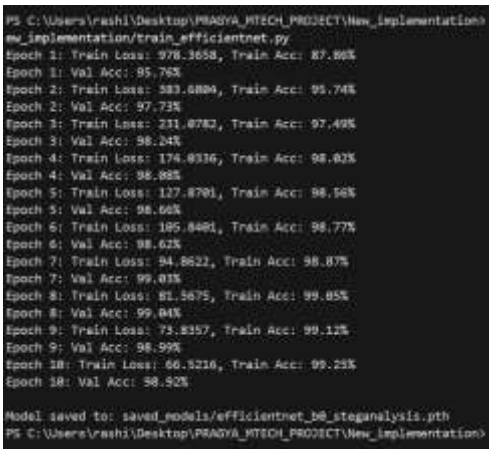

Fig 6: CNN Testing accuracy 99.52%

With very few misclassifications, the confusion matrix showed that the cover and stego images were nearly perfectly separated. The Area Under Curve (AUC) was quite near 1.0, as indicated by the steep ROC curve near the top-left corner.

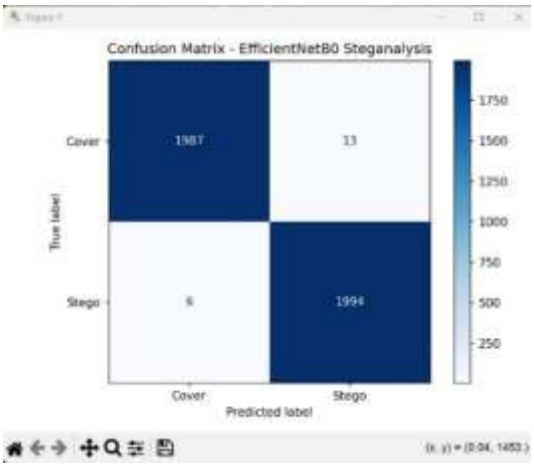

Fig 7: CNN Confusion Matrix

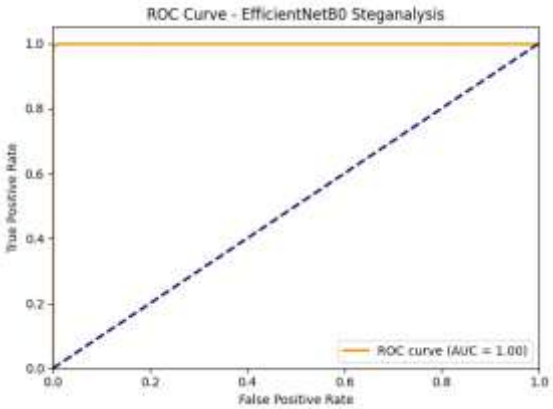

Fig 8: CNN ROC curve

Strong convergence with little overfitting was indicated by the training accuracy, which increased gradually to 98%. The reliability and efficiency of the model in capturing fine-grained steganographic properties are validated by these visuals.

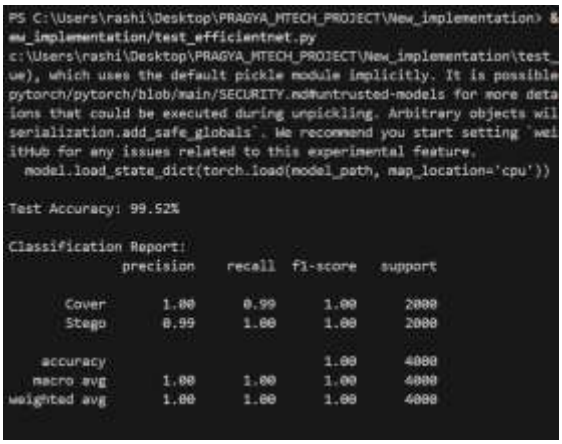

Fig 9: CNN Training model accuracy 98%

12.1.2 RNN (CNN + GRU)

The testing accuracy of the RNN model was 80.20%.

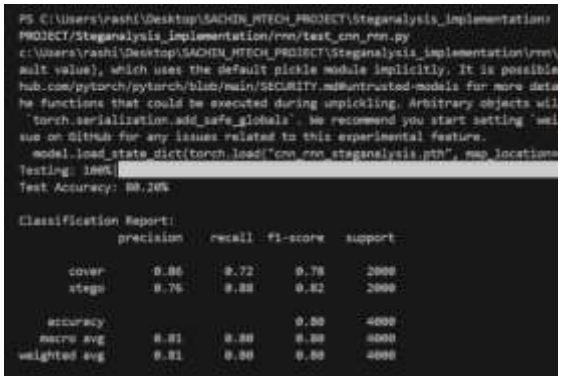

Fig 10: RNN Testing accuracy 80.20%

In comparison to CNN, the confusion matrix displayed a greater false negative rate, especially when it came to categorizing stego pictures. A moderate trade-off between true positive and false positive rates was reflected in the less steep ROC curve.

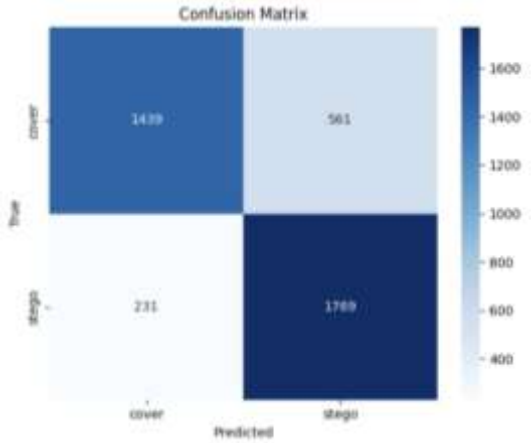

Fig 11: RNN Confusion Matrix

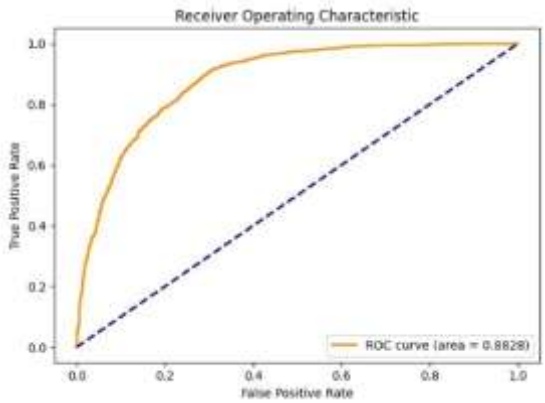

Fig 12: RNN ROC Curve

With a marginally slower rate of convergence than CNN, training accuracy reached a peak at about 79.78%. This suggests that although the sequential modeling was able to capture spatial dependencies, it had trouble with low-payload or ambiguous embeddings.

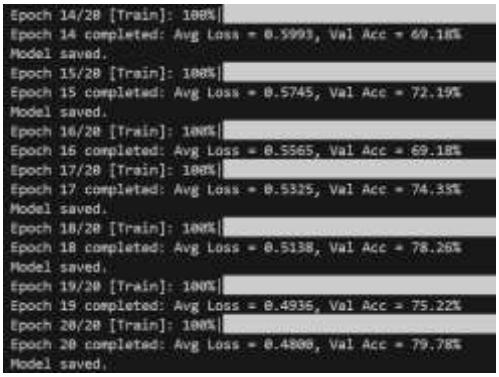

Fig 13: RNN Training accuracy 79.78%

12.1.3 GAN (Discriminator)

On the test set, the GAN discriminator's accuracy was 91.70%.

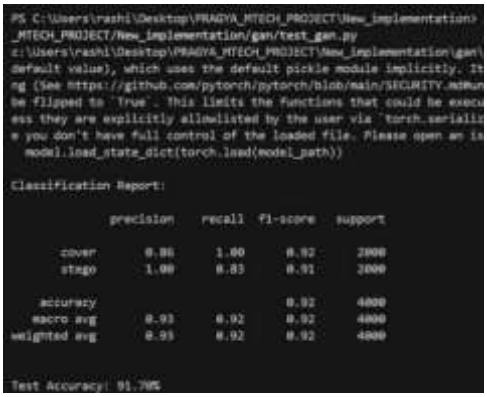

Fig 14: GAN Test accuracy 91.70%

Strong classification performance was shown by the confusion matrix; however, a few cover photos were incorrectly identified as stego, most likely as a result of overfitting to texture patterns. Although it deviated somewhat from ideal, the ROC curve showed strong separability.

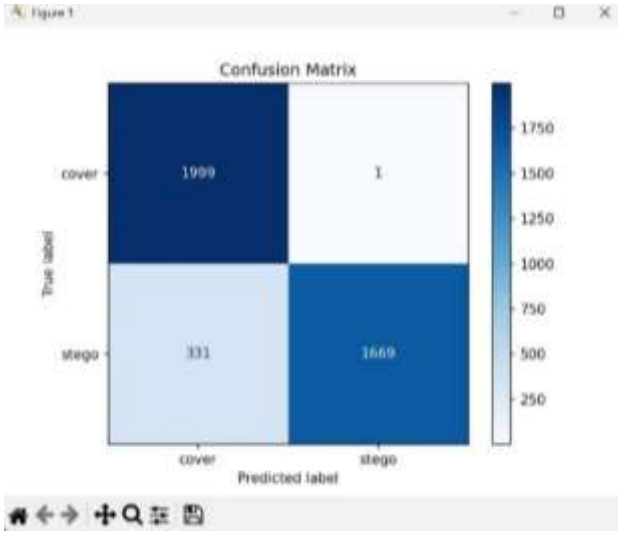

Fig 15: GAN Confusion Matrix

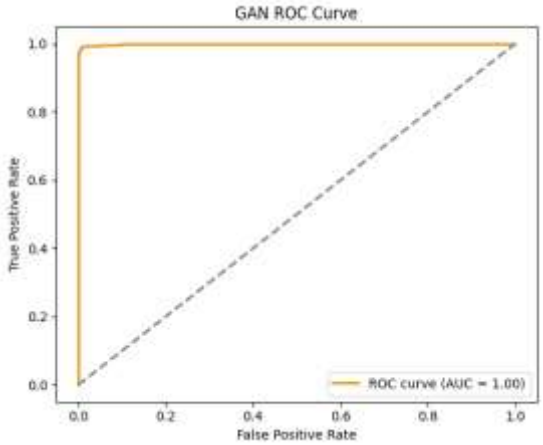

Fig 16: GAN ROC Curve

As is common in adversarial scenarios, training curves fluctuated moderately before stabilizing. These photos demonstrate how well the discriminator picks up on minute image disturbances, particularly in high-payload or warped stego images.

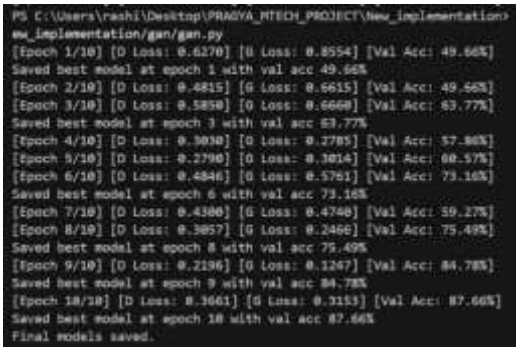

Fig 17: GAN Training model accuracy 87%

12.1.4 Traditional Models (Rich Model + LightGBM & MLP)

The accuracy of the MLP was 75% and that of the Rich Model with LightGBM was 75.5%.

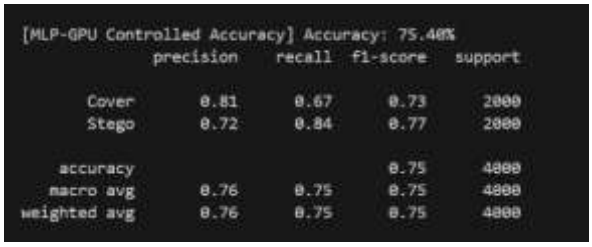

Fig 18: MLP Traditional Method Accuracy 75%

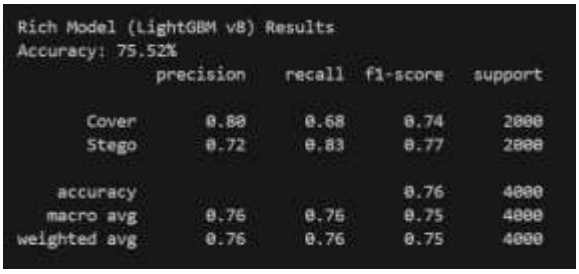

Fig 19: Rich Model Traditional Method Accuracy 75.5%

Due to feature restrictions, their confusion matrices showed symmetrical misclassification patterns, especially when it came to labeling clean cover images as stego.

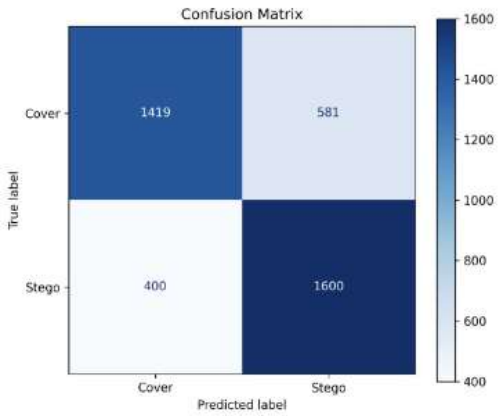

Fig 20: MLP Traditional Method Confusion Matrix

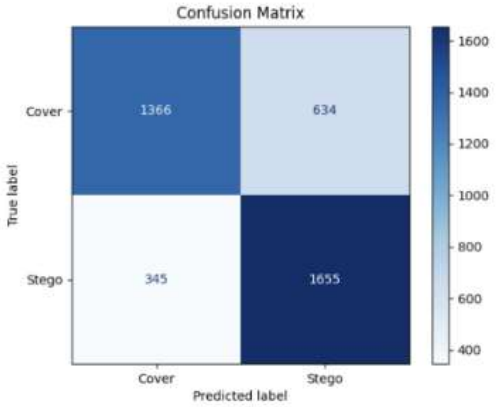

Fig 21: Rich Model Traditional Method Confusion Matrix

ROC curves displayed limited class separability and were less noticeable. Training was computationally light and accuracy curves were stable, confirming their usefulness as baseline models. Their visual metrics, however, highlight their poor capacity to adjust to low-payload fluctuations or more recent steganographic techniques.

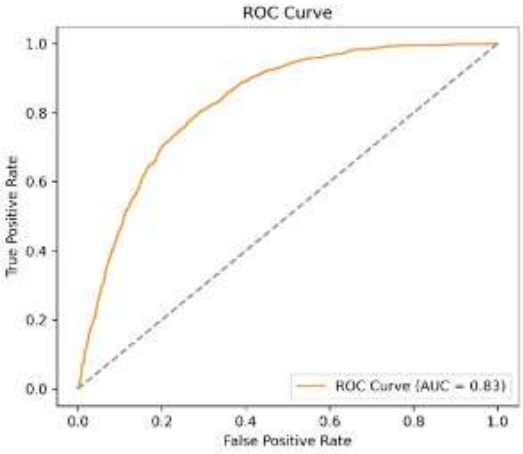

Fig 22: MLP Traditional Method ROC Curve

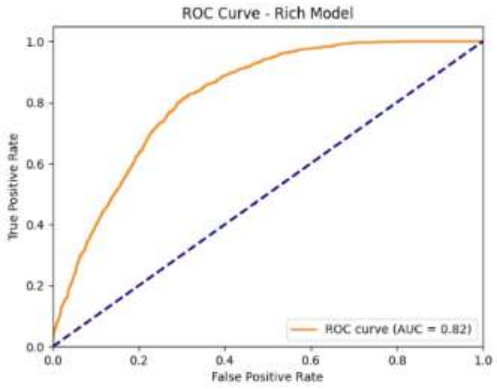

Fig 23: Rich Model Traditional Method ROC Curve

When taken as a whole, the visualizations support the conclusions drawn from Chapter 6. CNNs and other deep learning models performed better in classification, with low error margins and high confidence. GANs demonstrated competitive performance, especially in high-confidence circumstances. RNNs demonstrated limits in unclear settings, despite their value in sequence modeling. Conventional models, like LightGBM and MLP, provided accurate baselines but lacked learnt feature sensitivity. The deeper misclassification analysis covered in Section 7.2 is based on these observations.

11.2 Analysis of Misclassifications

Notwithstanding the excellent accuracy (varying from around 75% to 99.5%) attained by all five model types—CNN, RNN, GAN, Rich Model, and MLP—a number of misclassifications were observed during the testing session. Analyzing these misclassifications in detail shows how various model types react to intricate stego patterns and offers important insights into the advantages and disadvantages of each model each.

i. **Nature of Misclassified Samples:** The following traits were present in a large number of the incorrectly labeled photos:

- Regions with low contrast or visual smoothness, which provide the models with less discriminative artifacts to train from. The absence of high-frequency features makes categorization more difficult because steganographic embedding frequently affects them.
- Noisy or highly textured cover images, which the GAN discriminator and occasionally more conventional models like the Rich Model mistook for stego. The models were tricked by these textures into seeing natural noise as embedded signals.
- Even well-tuned models like CNNs or MLPs had trouble detecting stego traces since sophisticated adaptive steganography methods created minute pixel changes. The label "cover" was commonly applied to these samples.
- Both deep learning and conventional models had ambiguity due to boundary situations with partial or negligible payloads, particularly those that mostly depended on histogram-based features like MLP and LightGBM.

ii. Observed Patterns by Model Type

- CNN Misclassifications: On photos containing global low-frequency material, the CNN model occasionally failed because the localized steganographic noise was too faint. Especially in the early phases of training, these failures point to a restriction in the spatial resolution of features learned by convolutional layers.
- RNN Misclassifications: Because RNNs used sequential features that were taken from images (such flattened rows or learnt feature maps), they occasionally misclassified photos that did not include clearly steganographic sequential pixel-level changes. This implies that RNNs are more susceptible to the continuity and layout of the embedding noise.
- Some cover photos were incorrectly labeled as stego by the discriminator in the GAN model, particularly when there was natural noise or compression artifacts. This suggests that rather than using genuinely discriminative stego characteristics, the discriminator has a propensity to overfit to specific visual patterns.
- Rich Model (LightGBM) Misclassifications: Structured patterns in cover images that resembled typical steganographic markers (such as block borders or periodic textures) occasionally deceived this conventional

model. Due to its fixed feature set, it was susceptible to these artifacts and occasionally did not adjust to embedding tactics that had not been seen before.

- MLP Misclassifications: The MLP-based classifier occasionally misidentified cover photos with uneven illumination or textural shifts as stego because it was trained on handmade characteristics (HSV, LBP histograms). Its linear layer mappings might not be able to differentiate between real color distribution fluctuations and modest embedding noise.

iii. **Implications and Mitigation Strategies:** Despite the remarkable results that both deep learning and classical models have demonstrated in identifying steganographic information, the misclassifications of all five models highlight several common and model-specific weaknesses:

- Adversarially clean stego images,
- Both conventional and deep models are confused by cover images that imitate stego features (for example, by using compression or natural noise).
- Minimal pixel variation images provide little information for learned or handmade features to take advantage of.

To mitigate such errors in future work:

- Ensemble Learning: By combining predictions from CNNs, RNNs, GANs, Rich Models and MLPs, one can increase resilience and lessen the bias of individual models.
- Confidence-based filtering: Using output probabilities to exclude ambiguous predictions could preserve high accuracy in real-world systems.
- Hard case augmentation: Models may be trained with uncommon pattern variations or adversarially constructed stego-cover pairs to enhance generalization.
- Feature Fusion: Through hybrid modeling, combining deep embeddings (from CNNs/RNNs) with handmade features (used in MLP and Rich Model) may improve classification performance.

12. CONCLUSION

This research evaluates five steganalysis models—CNN (EfficientNet-B0), RNN (CNN-GRU hybrid), GAN-based discriminator, Rich Model with LightGBM, and MLP with handcrafted features—trained on a controlled color picture dataset from ALASKA2. Each model underwent thorough training and evaluation using a single pipeline that includes improved preprocessing, augmentation, and performance metrics.

Using spatially constrained artifacts, the CNN-based model identified stego content with 99.52% accuracy. The adversarially trained GAN-based discriminator was effective at detecting high-payload stego images, but it struggled to generalize to subtle embedding patterns. While the RNN-based model proved useful for describing geographical relationships, it was prone to payload sparsity and overfitting. Traditional models like MLP and LightGBM, albeit having lower accuracy (~75%), have strong interpretability and computational efficiency, making them suited for lightweight forensic applications.

Performance was visualized using ROC curves, confusion matrices, and training accuracy plots, which proved the superiority of deep learning algorithms, specifically CNNs. However, an in-depth misclassification examination discovered that all models have continuous difficulties when exposed to low-contrast, noisy, or boundary-case samples.

By developing a unified training and evaluation framework, this study provides an empirically grounded comparison of modern and traditional steganalysis techniques in color image domains, paving the way for hybrid and ensemble-based detection frameworks.

12.1 Limitations

i. Practical limits.

- Hardware Dependence: Deep models such as CNNs and GANs require high-end GPUs for optimal training, making deployment problematic in resource-constrained environments.
- Inference Latency: While CNNs are rather fast, GANs and RNNs introduce additional latency, making them unsuitable for real-time applications.
- Payload Agnosticism: Models are trained as binary classifiers (cover vs. stego) without explicit knowledge of payload size or embedding method, limiting forensic interpretability.

ii. Theoretical limitations.

- Limited dataset scope: ALASKA2 is used for all experiments. The lack of cross-dataset validation (such as BOSSBase and BOWS2) reduces the generalizability of results.
- Overfitting Risks: Even after data augmentation, RNNs and GANs demonstrated overfitting, especially in low-payload or high-frequency images.
- Lack of Explainability: None of the models currently employ XAI techniques such as Grad-CAM, SHAP, or LIME, restricting their applicability in legal or forensic evidence settings.

12.2 Future Work

i. Generalization across different domains and datasets: The current evaluation is limited to the ALASKA2 dataset, which allows for controlled embedding using LSB-based techniques. To mimic varied conditions, future research should expand the evaluation to include multiple datasets such as BOSSBase, BOWS2, Dresden Image Database, and Stego-App. This will allow exploration of:

- Cross-dataset domain adaptation techniques, such as Domain-Adversarial Neural Networks (DANN) and Maximum Mean Discrepancy (MMD) regularization, can enhance model generalization.
- Meta-learning approaches (e.g., MAML, Reptile) are used to create models that can quickly adapt to previously unknown embedding patterns while requiring minimal fine-tuning.
- Evaluation of payload distribution mismatch, where the training and testing datasets have different embedding payload sizes or steganographic methods.

ii. Development of Payload-Aware and Multi-Class Steganalyzers.

- Hugo, J-UNIWARD, WOW, and nsF5 are examples of multi-class classification systems that consider both the presence of steganographic information and the specific embedding method.
- Payload regression methods, like Support Vector Regression (SVR) or encoder-decoder networks, can be used to calculate payload density.
- Creating multi-task learning frameworks that include payload categorization, stego detection, and changing region localization in a joint optimization environment.

iii. Hybrid and ensemble architectures: Individual deep learning models, such as CNNs, RNNs, and GANs, offer distinct advantages, but ensemble approaches can employ complementary decision limitations. Future directions include:

- Stacked ensemble architectures used CNN feature maps, RNN sequential encodings, and GAN discriminator embeddings to create stacked representations.
- Attention-based fusion mechanisms (e.g., Transformer-based feature aggregation or self-attention pooling) dynamically assess feature importance across branches.
- Bayesian neural networks are employed in ensembles to represent uncertainty in steganalysis predictions.

iv. Applying Explainable AI (XAI) techniques:

- Gradient-weighted Class Activation Mapping (Grad-CAM) and SHAP values are utilized to identify which image elements impacted the steganalysis decision.
- Heatmaps of feature attribution over RGB channels are utilized to establish relationships between spatial/texture changes and stego prediction confidence.
- Explanation regularization, as a loss term, is used to force model decisions based on semantically acceptable regions, hence improving interpretability and robustness.

v. Transformer-based architectures and vision-language models.

- Emerging designs, such as Vision Transformers (ViTs), can be employed in steganalysis to simulate global interconnectedness and spatial attention across colour channels.
- ViT-based steganalysis backbones, trained with patch embeddings and positional encodings, may improve the detection of minor anomalies scattered across broad areas.
- Swin Transformers or PVTv2 variants are used to train hierarchical representations.
- Exploratory integration of multimodal models using transformer backbones conditioned on textual signals (e.g., metadata, captions) to detect visual anomalies.

vi. Resilience to adversarial and adaptive steganography.

- Adversarial training pipelines use PGD or FGSM to mimic evasive stego attacks and improve model resistance.
- In high-dimensional space, the margin of decision boundaries is expanded via defensive distillation and noise-injection regularization.
- To demonstrate robustness to disturbances, use known countermeasures such as randomized smoothing or interval bound propagation.

vii. Real-time, low-latency inference for field deployment.

- Weight pruning, knowledge distillation, and quantization (e.g., INT8, FP16) are model compression techniques that minimize model size and inference costs.
- ONNX or TensorRT pipelines are utilized for edge device deployment, including Jetson Nanos and mobile SoCs.
- Efficient neural architectures (e.g., MobileNetV3, EfficientNet-Lite) were retrained on stego datasets to enable real-time detection at high frame rates.

viii. Proposals to Detect Localized Stego Artifacts and Regions

- Using region proposal networks (RPNs) to separate anomalous patches.
- The usage of object detection backbones (such as YOLOv8 or Faster R-CNN) trained with stego-mask overlays.
- Use semantic segmentation models (such as DeepLabv3+) that have been trained to detect spatial anomalies caused by embedding errors.

13. REFERENCES

[1] Kuznetsov, A., Luhanko, N., Frontoni, E., & Rundo, L. (2023). Image steganalysis using deep learning models. Multimedia Tools and Applications, 83, 48607–48630. Springer. https://doi.org/10.1007/s11042-023-17591-0

[2] Tang, W., Li, H., Luo, W., & Huang, J. (2016). Adaptive steganalysis based on embedding probabilities of pixels. IEEE Transactions on Information Forensics and Security, 11(4), 734–745. https://doi.org/10 .1109/TIFS.2015.2507159

[3] You, W., Zhang, H., & Zhao, X. (2021). A Siamese CNN for image steganalysis. IEEE Transactions on Information Forensics and Security, 16, 291–306. https://doi.org/10.1109/TIFS.2020.3013204

[4] Kodovsky, J., Fridrich, J., & Holub, V. (2012). Ensemble classifiers for steganalysis of digital media. IEEE Transactions on Information Forensics and Security, 7(2), 432–444. https://doi.org/10.1109/ TIFS.2011.2175919

[5] Yang, H., He, H., Zhang, W., & Cao, X. (2021). FedSteg: A federated transfer learning framework for secure image steganalysis. IEEE Transactions on Network Science and Engineering, 8(2), 1084–1094. https://doi.org/10.1109/TNSE.2020.2996612

[6] Sahu, A. K., & Sahu, M. (2020). Digital image steganography and steganalysis: A journey of the past three decades. Computer Science – De Gruyter, 21(2), 157–172. https://doi.org/10.1515/comp-2020-0136

[7] Amritha, P. P., Sethumadhavan, M., Krishnan, R., & Pal, S. K. (2020). Anti-forensic approach to remove stego content from images and videos. Journal of Cyber Security and Mobility, 8(3), 295–320. https://doi.org/10.13052/jcsm2245-1439.831

[8] Zou, Y., Zhang, G., & Liu, L. (2019). Research on image steganography analysis based on deep learning. Journal of Visual Communication and Image Representation, 60, 266–275. https://doi.org/10. 1016/j.jvcir.2019.02.034

[9] Bashir, B., & Selwal, A. (2021). Towards deep learning-based image steganalysis: Practices and open research issues. In Proceedings of the International Conference on IoT-Based Control Networks & Intelligent Systems (ICICNIS 2021). SSRN. https://doi.org/10.2139/ssrn.3883330

[10] Pevný, T., Fridrich, J., & Ker, A. D. (2012). From blind to quantitative steganalysis. IEEE Transactions on Information Forensics and Security, 7(2), 445–454. https://doi.org/10.1109/TIFS.2011.2175918

[11] Avcibas, I., Memon, N., & Sankur, B. (2003). Steganalysis using image quality metrics. IEEE Transactions on Image Processing, 12(2), 221–229. https://doi.org/10.1109/TIP.2002.807363