INTERNATIONAL JOURNAL OF LATEST TECHNOLOGY IN ENGINEERING, MANAGEMENT & APPLIED SCIENCE (IJLTEMAS)

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

AI-Based Steganography Method for Iot Data Communication Using Secure Blockchain

Kavyashree L, KH Sandeep Reddy, A. M. Giri Prakash Reddy, Basava Reddy Hampi, Karthik R Raghothama

Department of CSE, DSATM, Bengaluru, Karnataka, India

DOI: https://doi.org/10.51583/IJLTEMAS.2025.140600075

Abstract— As the demand for secure data transmission continues to rise, steganography has become an essential technique in information security. Despite its importance, conventional steganographic approaches often lack flexibility and robust content protection. This paper introduces an innovative method that combines artificial intelligence with adaptive steganography and real-time content safeguarding. The proposed framework intelligently tailors steganographic strategies by analyzing the type of cover media and assessing potential risks. It also incorporates mechanisms for dynamic content modification, which respond to user interactions or predefined conditions. This work outlines the architecture, development, and performance assessment of the system. Experimental results highlight the benefits of AI-driven adaptability and dynamic content control in strengthening steganographic techniques. Additionally, the paper explores current challenges, future enhancements, and practical use cases of this forward-looking methodology.

Index Terms— Steganography, Artificial Intelligence (AI), Contrastive Learning, Generative Adversarial Networks (GANs).

I. Introduction

The growing necessity for secure communication channels has placed steganography at the forefront of modern data protection methods. While conventional steganographic techniques provide basic concealment, they often fall short in terms of flexibility and robust content safeguarding. To address these limitations, this paper introduces a cutting-edge solution that integrates artificial intelligence into adaptive steganography. By using AI, the system intelligently refines embedding techniques depending on the characteristics of the cover media and the nature of potential threats. This paper explores the architecture, development, and effectiveness of this AI-integrated steganographic model, demonstrating its suitability for today's evolving digital security requirements.

A. Role of AI in Modern Steganography

Steganography, the science of embedding secret messages within ordinary digital content, has faced challenges in concealment efficiency, resilience, and adaptability across various media types. With the emergence of artificial intelligence—especially deep learning—a new path has opened for overcoming these challenges. Deep learning algorithms, trained on vast datasets, can identify intricate patterns and relationships within data, enabling them to evaluate media files such as images, audio, or videos with exceptional precision.

This allows for a deeper understanding of a media file's inherent properties, such as statistical behavior and noise distribution. Leveraging this intelligence, more advanced embedding methods can be designed to enhance both the durability and effectiveness of hidden data. AI brings solutions to key issues in traditional steganographic systems, including:

Improved Security: AI models can enhance the secrecy and imperceptibility of embedded data by customizing the embedding strategy based on the specific media file's features. They can also generate stego-content that remains indistinguishable from the original, even under statistical scrutiny, mitigating the risk of detection by steganalysis tools.

Increased Payload Capacity: Neural networks can be used to boost the volume of data embedded without compromising visual or auditory quality. This is achieved by adjusting the embedding method in real-time based on the frequency and spatial domain features of the media.

Better Resilience to Attacks: AI helps in building steganographic methods that can withstand various forms of manipulation, such as compression, cropping, and noise insertion, which are common in hostile environments.

B. Deep Learning in Media Content Analysis

Deep learning, a subfield of AI, has revolutionized the way digital content is analyzed across numerous industries—including steganography. Unlike traditional methods that rely on manually crafted features, deep learning models learn intricate representations directly from large datasets. When trained on thousands or millions of media samples and their stego counterparts, these models develop the ability to detect subtle variations caused by data embedding.

In the realm of steganography, deep learning allows the system to "fingerprint" each media file, understanding its structural nuances and statistical behaviors. This knowledge becomes pivotal for the adaptive embedding phase, wherein the AI tailors the embedding process specifically to the media file's unique properties—ensuring that the hidden data remains undetectable.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

Instead of relying on static embedding rules, the deep learning model dynamically chooses the most effective approach for each instance. Its ability to learn end-to-end enhances performance significantly, as it can simultaneously optimize both message extraction and embedding processes.

Generative Adversarial Networks (GANs) further elevate this approach by generating synthetic stego-content that mirrors the statistical characteristics of genuine media. This makes detection by even the most advanced steganalysis tools significantly more difficult.

Deep learning's adaptability allows these models to evolve with new media types, such as high-resolution video and 3D content, ensuring their effectiveness in future digital ecosystems. As a result, AI-powered steganography remains viable and secure, even as technology and media consumption trends continue to shift.

In conclusion, integrating deep learning into steganographic systems significantly enhances traditional data hiding techniques and sets the stage for future innovations in secure digital communication. As these models grow more refined and are trained on larger datasets, the potential for truly undetectable communication expands—marking a transformative step in the field of information security.

Project Scope and Motivation

The continuous demand for trustworthy and secure communication methods, particularly in environments where confidentiality is essential, forms the core motivation behind this project. Traditional steganographic approaches, though capable of concealing information, often struggle with several limitations—such as low data hiding capacity, vulnerability to various forms of attacks, and poor adaptability across diverse media types. These constraints restrict their effectiveness in real-world applications.

This project seeks to overcome these barriers by leveraging the capabilities of artificial intelligence, specifically deep learning, to build a more robust and intelligent steganographic framework. The proposed AI-driven steganography system introduces dynamic protection and adaptability into the message hiding process, enhancing both reliability and security.

The system utilizes two specialized deep learning models, each fulfilling a unique role:

Adaptive Embedding Model: This model is trained to perform an in-depth analysis of the cover media, identifying optimal regions and methods for embedding data while preserving the visual or auditory integrity of the content.

Content Protection Model: Focused on maintaining the confidentiality of the hidden message, this model continuously evaluates the stego-media for potential threats. It is capable of detecting and neutralizing attacks aimed at uncovering or altering the embedded data in real-time.

Together, these components aim to create a self-monitoring steganographic system that adjusts its techniques dynamically based on the nature of the media and any emerging security threats.

The successful implementation of this AI-enhanced approach could lead to significant advancements in the field, making steganography more practical and resilient for applications such as:

- **Secure communication** in environments where data privacy is paramount,
- **Digital copyright protection** for media content,
- **Covert information exchange** in scenarios with limited computational resources.

Ultimately, this project aspires to introduce a flexible, intelligent, and secure method for hiding sensitive information—well-suited to the evolving demands of the digital age.

Proposed Work

A. Data Acquisition

To train an AI model capable of advanced steganography, it is essential to start with a comprehensive and diverse dataset. Our approach involves collecting a large volume of cover images across multiple categories and formats to represent various media types. Alongside these, we will gather a range of encrypted or confidential messages in image format to simulate real-world hidden data scenarios.

Additionally, to better understand effective embedding strategies and enhance resistance to detection, we will include stego-images generated using multiple steganographic methods. In certain experiments, user feedback regarding the visibility of hidden messages will be incorporated to provide insights into perceptibility. We will also compile datasets related to steganalysis—the detection of hidden messages—to ensure the AI system learns from both concealment and detection strategies. This extensive and varied data pool will empower the AI model to learn optimal, covert embedding techniques.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

B. Text Embedding Generation

Embedding text accurately and securely into digital content is a crucial part of the steganographic process. We utilize **Word2Vec**, a well-established neural model in Natural Language Processing (NLP), to convert textual information into numerical vectors that machines can understand and manipulate.

Word2Vec operates using two main architectures: **Skip-gram** and **Continuous Bag of Words (CBOW)**, both of which produce meaningful, dense vector representations by analyzing context. These vectors preserve the semantic essence of each word.

The text embedding process involves the following steps:

Model Preparation: Either a pre-trained Word2Vec model is adopted, or a new one is trained using a domain-specific text corpus to create contextually rich word representations.

Text Tokenization: The secret message is broken down into individual tokens (words), each of which is converted into its corresponding vector using the Word2Vec model.

Message Vector Construction: To form a unified message representation, these vectors are aggregated. This can be achieved through techniques such as averaging, or more advanced methods like **Principal Component Analysis (PCA)** or **autoencoders** to reduce dimensionality and extract deeper semantic meaning.

These vector-based embeddings are well-suited for integration into media files, enhancing both the stealth and resilience of the steganographic process.

C. Least Significant Bit (LSB) Steganography

One of the simplest yet highly effective techniques for image-based data hiding is **Least Significant Bit (LSB) steganography**. This method embeds hidden information, such as text embeddings, into the least significant bits of pixel values within an image, ensuring minimal distortion.

First, the vectorized message is translated into a binary format. Then, this binary sequence is embedded into the cover image by modifying the lowest bit(s) of the pixels, effectively inserting the secret data without noticeably altering the image.

This method is favored for its simplicity, high capacity, and excellent invisibility when implemented correctly. However, it may be susceptible to image processing operations unless enhanced with more intelligent embedding techniques.

D. Integration of Generative Adversarial Networks (GANs)

To push the boundaries of steganography, we integrate **Generative Adversarial Networks (GANs)** into the system to enable dynamic and adaptive content protection. A GAN consists of two neural networks: a **generator**, which creates new stego-images with hidden messages, and a **discriminator**, which attempts to differentiate between cover and stego-images.

Through continuous adversarial training, the generator learns to embed messages in such a way that even the discriminator—designed to detect them—fails to recognize them. As the generator refines its embedding techniques, it begins subtly manipulating textures, colors, and other image attributes to hide data more seamlessly.

This adversarial feedback loop results in highly sophisticated steganographic methods that are not only hard to detect with the naked eye but also resistant to automated detection algorithms. The evolving battle between the generator and discriminator ensures constant innovation in hiding strategies and strengthens the overall security of the system.

Proposed Steganography Framework Using GANs

This work introduces a generative adversarial network-based steganography system that allows seamless concealment of diverse message types, including text, audio, and image data, within standard digital images. The system is engineered to maintain high fidelity between the stego and the original image, while enabling reliable extraction of the embedded content. The architecture has been divided into several critical modules: preprocessing, encoding, message integration (embedding), decoding, adversarial training, and evaluation. Each of these components is meticulously designed to ensure the dual goals of imperceptibility and recoverability.

1. Input Preprocessing and Standardization

Before any message can be hidden within an image, it must be transformed into a format compatible with the GAN model. Each message type undergoes a specific preprocessing pipeline that standardizes its structure and dimensionality.

For **textual data**, natural language inputs are passed through embedding mechanisms like the BERT model or Transformer-based sentence encoders. These models convert variable-length text into fixed-length, dense vector embeddings that preserve semantic information.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

In the case of **audio data**, the raw waveform is first segmented and transformed into a spectrogram—a visual representation of the frequency content over time. Alternatively, features such as MFCCs (Mel Frequency Cepstral Coefficients) or embeddings from pretrained audio encoders (like VGGish) are used to compress the audio into fixed-size representations.

When **images** are used as the secret message, they are resized to a standard resolution and normalized across channels. Image-specific enhancements such as color normalization or histogram equalization may also be applied to ensure visual consistency and compatibility with the embedding network.

2. Feature Extraction Through Encoding Networks

Once the input data is standardized, it is passed through a feature encoding stage that reduces the high-dimensional input into a latent representation. This stage is crucial as it creates a compact form of the message that is suitable for embedding.

Each message type utilizes a domain-specific encoder. For text, models like GRU-based RNNs or Transformer encoders are employed to distill the sentence vector. For audio, a convolutional encoder followed by a recurrent unit may be used to capture temporal dependencies. When working with image data, deep CNNs such as ResNet or DenseNet are applied to generate a deep latent feature map.

These encoded representations form the backbone of the message that will be hidden within the cover image during the embedding process.

3. Embedding Mechanism: Message Injection into Cover Image

The core component of this system is the generator network, which performs the embedding task. This network is responsible for integrating the encoded message into the cover image such that the final stego image remains visually indistinguishable from the original.

Typically, architectures like U-Net or encoder-decoder models with skip connections are used as the base structure of the generator. The message features are spatially concatenated or added into the bottleneck or intermediate layers of the generator. This allows fine-grained control over how the message is injected into various frequency components of the image.

To ensure that the stego image retains the quality of the original, the loss function guiding this module includes multiple terms. A **pixel-wise loss** penalizes significant deviations from the cover image at a low level. A **perceptual loss** (using VGG features) ensures that high-level content and style remain unaffected. Most importantly, an **adversarial loss** from the GAN discriminator is used to promote realism, forcing the generator to produce images that look natural and unaltered.

4. Discriminator: Detecting Stego vs. Cover

The discriminator serves as the adversarial counterpart to the generator. It is trained to differentiate between untouched cover images and stego images that contain hidden information. The goal of the generator is to produce stego images that are indistinguishable from real images, while the discriminator tries to classify them correctly.

To enhance sensitivity, the discriminator may employ a **PatchGAN** structure, which evaluates the realism of small patches of the image rather than the entire image. This helps capture high-frequency artifacts or local inconsistencies introduced during the embedding phase. In some implementations, multiple discriminators at different scales are used to enforce both local and global realism.

The adversarial feedback from the discriminator is used to iteratively refine the generator, leading to increasingly subtle and undetectable stego images.

5. Message Recovery via Decoder Networks

After embedding, it is vital that the hidden message can be successfully extracted from the stego image. This task is handled by the decoder module, which operates as the inverse of the embedding pipeline.

The decoder receives the stego image and attempts to reconstruct the original message from it. Depending on the message type, specialized decoders are used. Text decoders often utilize RNN-based architectures, while image decoders use convolutional transposed layers or upsampling blocks. For audio, a combination of CNNs and recurrent layers are employed to reconstruct the spectrogram or waveform.

A reconstruction loss (e.g., mean squared error or binary cross-entropy) is used to guide the decoder during training. The objective is to minimize the discrepancy between the original and retrieved messages, ensuring high accuracy even after embedding.

6. Adversarial Training Pipeline

The system uses a GAN training approach where the generator and discriminator are updated in a competitive fashion. The generator strives to create high-quality stego images that retain the message and deceive the discriminator, while the discriminator aims to become better at detecting embedded content.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

The overall training loss is a composite of several components:

- **GAN Loss**: Ensures the stego images are visually realistic.
- **Message Loss**: Penalizes errors in the recovered message.
- **Perceptual Loss**: Encourages similarity in high-level features.
- **Embedding Regularization**: Prevents overfitting and ensures the encoded message doesn't dominate the image features.

Training proceeds in alternating steps, typically using Adam or RMSprop optimizers, with careful tuning of learning rates and loss weights to maintain equilibrium between the networks.

7. System Evaluation: Robustness, Quality, and Security

To validate the effectiveness of the system, multiple performance metrics and test scenarios are applied.

Imperceptibility is evaluated using standard image similarity metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM). Higher values in these metrics indicate minimal deviation from the cover image.

Message Fidelity is assessed by comparing the extracted message with the original input. Depending on the type, accuracy metrics, bit error rates (for binary data), or mean squared error (for continuous data) are calculated.

Robustness Testing involves subjecting the stego image to various distortions—like compression (JPEG), noise addition, blurring, resizing, or cropping—and then evaluating the success rate of message extraction post-distortion.

Steganalysis Resistance is tested using modern steganalysis tools or deep-learning-based classifiers. These tools attempt to detect stego images by identifying statistical anomalies. A low detection rate implies high security.

8. Regularization and Augmentation for Real-World Resilience

During training, the model is exposed to a wide array of augmentations to improve generalization. These include:

- Random horizontal and vertical flips
- Cropping and resizing
- Gaussian noise and salt-and-pepper distortions
- Simulated compression artifacts

Regularization techniques such as dropout, weight decay, and data randomization are also employed to prevent overfitting and improve performance on unseen images.

9. Flexibility and Modular Expansion

The modular architecture of the system ensures it can be extended for future applications. New message types can be added by integrating corresponding encoders and decoders. Moreover, support for **video steganography** or QR code-based message embedding can be incorporated by extending the preprocessing and decoder modules.

The system is also adaptable to various cover media formats—potentially including 3D models or point clouds in extended applications.

Implementation of The Framework

A. Environment Setup and Development Platform

To facilitate the training and experimentation of deep learning models, this project leverages a cloud-based development environment provided by Google Colab. This platform eliminates the need for setting up local environments by offering a ready-to-use infrastructure with popular machine learning libraries such as TensorFlow and Keras pre-installed.

A key benefit of using Google Colab is its provision of free GPU resources, which significantly accelerates the training of computationally intensive deep learning models. By simply initiating a new Jupyter Notebook within the Colab interface, users can begin development by importing required libraries and dependencies. This notebook serves as the central workspace for defining model architecture, loading and preprocessing data, and conducting training routines for the adaptive steganographic system.

Through this cloud-native setup, researchers can streamline their workflow while ensuring the system is both scalable and reproducible without any hardware constraints.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

B. Data Preparation and Processing

Preparing the input data is a foundational step in constructing a robust steganographic system. Data preprocessing ensures the input conforms to the expected format, structure, and quality required by the neural networks for effective training and inference.

1) Sourcing Cover Image Data

The first stage involves curating a comprehensive and varied dataset of cover images. These images act as the base carriers for hidden messages. To ensure generalizability and broad coverage, the dataset includes photographs from a variety of categories such as natural landscapes, urban scenes, human portraits, and object images.

Important criteria during selection include high color fidelity, consistent resolution, and minimal visual noise or compression artifacts. This ensures that embedded data does not distort visibly or degrade under real-world conditions.

2) Preparing Cover Images for Model Input

Once the dataset is assembled, preprocessing techniques are employed to ready the images for training within the GAN framework. Images are resized to a fixed resolution compatible with the input layer of the neural network. Maintaining uniformity in image dimensions ensures stable training and efficient GPU memory utilization.

To augment the dataset and simulate real-world scenarios, image transformations such as horizontal flipping, random cropping, rotation, and brightness variation are introduced. These augmentations enhance the model's ability to handle diverse and unpredictable inputs during testing and deployment.

3) Converting Image Formats for Lossless Fidelity

Because steganographic accuracy relies on preserving pixel-level data, all input images must be converted to a lossless format—specifically PNG. JPEG files, which use lossy compression, can degrade embedded data due to irreversible changes in pixel values.

The conversion process begins by loading each JPEG image and decoding it into raw bitmap pixel data. This pixel matrix is then re-encoded using PNG's lossless compression algorithm. Once converted, these PNG images are stored in a designated directory for further processing. This guarantees the preservation of fine pixel details, making them suitable for message embedding and subsequent recovery.

This stage ensures consistency across the dataset and provides a clean, standardized input pipeline for downstream embedding and decoding operations.

C. Text Generation and Semantic Embedding Creation

A crucial component of the steganographic system is the handling of textual content, especially when messages are embedded within images. This section outlines the methodology for generating textual data and converting it into embeddings suitable for integration into cover images.

1) Generating Textual Data for Embedding

To simulate a broad range of message types that may be embedded, the system generates synthetic textual content. This is achieved by sampling from predefined corpora or phrase banks, which include commonly used sentences, domain-specific vocabulary, or user-defined messages.

This synthetic message generation introduces variability into the training data, enhancing the system's ability to handle real-world scenarios where the nature of embedded messages may differ drastically.

2) Converting Text to Embeddings Using Word2Vec

Once text is generated, it must be numerically represented to be processed by neural networks. This is accomplished using Word2Vec—an NLP model that translates text into high-dimensional vector representations.

Word2Vec uses two architectures—Continuous Bag of Words (CBOW) and Skip-gram—to learn semantic relationships between words based on contextual similarity. Words used in similar contexts are mapped to vectors that are spatially close in the embedding space.

The process includes:

Corpus Formation: A substantial body of text, either general or application-specific, is prepared as training material for the Word2Vec model.

Model Training: Based on the chosen architecture (CBOW or Skip-gram), Word2Vec is trained to predict words or their context using backpropagation and optimization techniques.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

Vector Generation: Upon training, the model produces dense, fixed-length embeddings for each word or phrase, capturing semantic and syntactic relationships.

These embeddings are then passed to the steganographic module for integration into the cover images.

3) Integration into the Steganographic System

The numerical embeddings serve as the encoded message inputs for the GAN. They are injected into the generator model, either through direct concatenation with the cover image feature map or through modulation within the network layers.

The system is trained to ensure that these embeddings can be both seamlessly hidden and later retrieved with high accuracy—balancing concealment and recoverability.

E. Dataset Construction and Partitioning

Following preprocessing and embedding, the system constructs training-ready input samples by forming pairs of cover images and their corresponding encoded messages.

Each training instance consists of:

- A PNG-format cover image that has undergone normalization and augmentation.
- A message embedding derived from the generated textual data using Word2Vec.

The dataset is split into training, validation, and optional testing subsets. The **training set** is used to optimize the model parameters, while the **validation set** monitors generalization and prevents overfitting. If a test set is included, it is reserved for final performance evaluation on unseen data.

This structured approach to dataset preparation ensures that the GAN is trained in a systematic and controlled manner, with continuous feedback to fine-tune performance.

E. Blockchain Integration for Secure Data Logging and Verification

To enhance the security and integrity of the steganographic communication system, a blockchain-based verification mechanism is integrated. This ensures that each message embedding and corresponding image can be cryptographically verified, audited, and traced without relying on a centralized trust model.

1) Blockchain Platform and Smart Contract Deployment

The project utilizes the Ethereum-based Sepolia testnet to simulate a secure blockchain environment. A custom smart contract is developed and deployed to the network, enabling the system to immutably record hash values of encoded images and their corresponding message metadata. The smart contract exposes functions to: Register a new encoded image by storing its SHA-256 hash and associated metadata (e.g., timestamp, sender ID). Retrieve historical logs for verification purposes. Verify image authenticity by comparing computed hashes against stored records.

2) Hash Generation and Blockchain Recording

After an image has been encoded with the message embedding, a SHA-256 hash of the final PNG file is computed locally. This unique fingerprint guarantees content integrity and enables tamper detection.

This hash, along with relevant metadata, is then sent to the deployed smart contract using Web3.py, a Python library for interacting with Ethereum. The transaction is signed using a private key and broadcasted to the Sepolia network, where it is validated and added to the ledger. Example metadata fields include: Image hash Timestamp Sender ID Message length Embedding vector dimensionality

3) Retrieval and Verification

During the decoding or validation phase, the system recomputes the hash of a received image and queries the blockchain to ensure it matches a previously recorded entry. Any mismatch indicates tampering or data corruption.

This integration ensures the confidentiality, authenticity, and non-repudiation of steganographic communications. The decentralized nature of blockchain eliminates single points of failure, and the immutable ledger enables transparent auditing for forensic or compliance purposes.

By combining deep learning-based steganography with blockchain-backed verification, the system delivers a powerful framework for secure and trustworthy hidden data communication—particularly suited for sensitive IoT and digital forensics applications.

Module Workflow

This project primarily focuses on developing an adaptive steganography system by training a deep learning model based on a Generative Adversarial Network (GAN).

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

Model Architecture Specification:

This initial phase involves structuring the architecture of the GAN, which includes two main components: the generator and the discriminator. The generator is designed to produce stego-images—images that embed secret messages subtly. It generally includes convolutional layers for efficient image processing, and it may use transposed convolutions to reconstruct image dimensions. The discriminator acts as a classifier that distinguishes between real cover images and those generated by the generator. Its design might mirror the generator in structure but is trained to identify embedded manipulations. This step involves deciding layer types, connections, and how the networks interact.

Dataset Preparation and Preprocessing:

This module addresses data ingestion and formatting for training. It involves loading cover image datasets and potentially embedded message datasets, which may already be transformed into embeddings—compact vector representations of text or data. Such embeddings play a critical role in concealing information. Standard preprocessing steps such as normalization, tensor conversion, and format compatibility for the chosen deep learning library (e.g., TensorFlow/Keras) are handled here.

Constructing and Compiling the GAN:

Once the architecture and data are in place, the model is constructed by implementing the generator and discriminator using the predefined structure. These models are compiled with selected loss functions and optimizers, which guide training. The loss function evaluates model performance, while the optimizer adjusts model weights to improve accuracy during training.

Adversarial Training Loop Execution:

At the heart of the system is the adversarial training loop. Training occurs in cycles (epochs), where each cycle alternates between updating the discriminator and generator. The discriminator is trained with both genuine images and stego-images, learning to classify them accurately. The generator is then trained to improve its ability to produce stego-images that deceive the discriminator. This creates a feedback loop, enhancing the quality and realism of generated images.

Model Assessment and Fine-Tuning:

After several training epochs, the system's output is evaluated. Metrics such as Peak Signal-to-Noise Ratio (PSNR) or manual review are used to assess how well the stego-images hide the embedded messages. The model's robustness against steganalysis (detection techniques) may also be tested. Based on these evaluations, adjustments are made to network design or training parameters to improve performance and concealment efficacy.

A. Module Implementation

Model Architecture Setup:

The architecture uses a GAN-based framework to conceal text embeddings within images. It defines the structure for the generator and discriminator—the core units responsible for embedding and detecting hidden information.

Generator Design:

The generator receives an RGB image (shape: 224x224x3) and a text embedding (shape: 1x128). The embedding is reshaped and spatially extended using a ZeroPadding2D layer to match the image's spatial dimensions. These inputs are then merged via concatenation. The result undergoes several Conv2D layers to capture interactions between the image and text. These layers are followed by batch normalization and LeakyReLU activations to enhance learning stability. Conv2DTranspose layers upscale the features back to original image dimensions, ensuring the output (a stego-image) retains visual integrity. The final output is a 3-channel RGB image indistinguishable from the original.

Discriminator Design:

The discriminator also receives an image and its corresponding embedding. Similar preprocessing and concatenation are performed. It applies convolutional layers with LeakyReLU activations and dropout for better generalization. The output is flattened and passed through a dense layer with a sigmoid function to classify whether the input is a real or stego image.

Adversarial Integration:

Both models are merged into a single adversarial setup. The discriminator improves in identifying subtle manipulations, while the generator learns to hide text with minimal detectable changes. This interaction enhances both security and effectiveness.

Loading and Preprocessing Data:

Functions are written to manage data loading, which includes retrieving cover images and preprocessed embeddings. These routines standardize image formats, scale pixel values, and convert them into tensors suitable for training. The project also emphasizes converting JPG images to PNG format

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

Lossless Quality: PNG ensures image data remains intact, which is crucial for embedding details.

Support for Transparency: Useful in image processing tasks involving overlays.

Sharper Rendering: PNG handles edges and text better than JPG, minimizing visual distortions.

The conversion is executed using Python's Pillow (PIL) library:

Load each JPG file via Image.open()

Save it in PNG format using image.save('filename.png'), ensuring no data degradation.

The process is typically automated for batch conversion.

Model Compilation:

After defining the generator and discriminator, each model is compiled with respective loss functions and optimizers using Keras. For the generator, a loss such as mean squared error (MSE) encourages it to produce visually similar stego-images. The discriminator, on the other hand, might use binary cross-entropy to classify inputs effectively. The Adam optimizer is often chosen for its efficiency in deep learning scenarios.

Adversarial Training Routine:

A loop drives the model's training. It begins by training the discriminator with actual and generated image pairs, adjusting its weights based on how well it distinguishes real from fake. Next, the generator is trained to improve its outputs such that they increasingly fool the discriminator. Gradients are calculated and applied back to improve the generator. This alternating update cycle continues for multiple epochs, progressively enhancing performance.

Saving and Loading Models:

Post-training, the model's weights, architecture, and optimizer state are saved for future reuse. Keras supports formats like HDF5 for this purpose. This enables continued training, deployment, or inference without retraining from scratch.

Stego-Image Generation:

The trained generator can now be used to embed messages into new cover images. A function accepts an image and a text embedding, applies the generator, and produces a stego-image. Before saving, outputs may be scaled or normalized to maintain image fidelity and match expected input ranges for downstream applications.

Experimental Setup and Results

The experimental setup outlines the resources, environment, and methodologies used for developing and evaluating the lane identification framework on Google Colab. The key components of the experimental setup are as follows:

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 demanding computational and processing requirements of deep learning models were addressed through a robust software environment in which this project was implemented. The primary software components are outlined below:

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

Programming Language: Python 3.8 was chosen for all scripting and model development due to its extensive library support and ease of integration with various data processing and machine learning frameworks.

Deep Learning Framework: The neural network models were built and refined using TensorFlow 2.4 and Keras 2.4.3. TensorFlow provided the necessary tools for model construction, while Keras offered a user-friendly API for rapid prototyping.

Supporting Libraries:

NumPy: Used for efficient numerical computations and data manipulation.

OpenCV: Employed for image processing tasks, including reading, writing, and transforming images during data preprocessing.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

Matplotlib: Utilized for visualizing training metrics and results, aiding in the interpretation of model performance.

Pandas: Used for handling and analyzing data efficiently.

Scikit-Learn: Leveraged for preprocessing data and evaluating model performance through a variety of metrics.

Version Control: Git was employed for version control, with the project hosted on GitHub to manage updates and enable collaborative development.

C. Dataset

The Oxford Flowers 102 dataset from Kaggle was selected for this project. It contains 8,189 images of flowers distributed across 102 categories, with each class representing a different flower type. This dataset is ideal for model training, providing a large, yet manageable set of images within a single category, allowing the models to focus on subtle distinctions between flower types.

D. Model Training

1. Architecture:

The generator is designed as a convolutional neural network (CNN) that incorporates both convolutional and deconvolutional layers. It takes two inputs: the image and its corresponding text embedding. The text embedding, initially a 1x1x128 vector, is reshaped and zero-padded to match the spatial dimensions of the image. The image and embedding are concatenated and passed through several layers of the network:

Convolutional Layers: The concatenated input is processed through Conv2D layers using LeakyReLU activation and BatchNormalization, with Dropout layers to regularize the model.

Deconvolutional Layers: Conv2DTranspose layers are used to upsample the feature maps, reconstructing the steganographic output image. These layers also include BatchNormalization and LeakyReLU activation, with the final layer using a sigmoid activation to produce the output image.

2. Training Procedure:

The generator is trained as part of an adversarial framework, working alongside a discriminator model. The discriminator's role is to differentiate between real and generated images, providing feedback to guide the generator's learning. The training procedure is as follows:

Loss Function: Binary cross-entropy is used as the loss function, reflecting the adversarial nature of the interaction between the generator and the discriminator.

Optimization: The Adam optimizer with an exponentially decaying learning rate is employed to update the generator’s parameters based on feedback from the discriminator and the need to minimize the loss.

Training Dynamics: During training, the model alternates between updating the discriminator with real and generated images and training the generator to better deceive the discriminator. The discriminator is trained on batches of real images with text and generated images with the corresponding embeddings. The generator’s weights are updated to improve its ability to confuse the discriminator.

3. Hyperparameters:

- **Batch Size:** Set to 20 to ensure detailed gradient updates for each training instance.
- **Epochs:** Training is conducted over 100 epochs to ensure sufficient learning and model refinement.

E. Model Evaluation

- **Generator Accuracy:** 0.96
- **Discriminator Accuracy:** 0.97
- **Decoder Model:**
    - **Decoder Accuracy:** 0.95
    - **Precision:** 0.9458
    - **Recall:** 0.7601
    - **F1 Score:** 0.845

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

F. Performance Metrics:

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

Conclusion and Future Scope

In this work, we have successfully developed a pipeline for embedding textual information within images using deep learning techniques. Beginning with a dataset of images, we extracted relevant features and employed a contrastive model to generate embeddings. These embeddings were then input into a GAN framework, alongside images, to produce stego-images containing the embedded text. Our experiments demonstrate the viability of leveraging neural networks for text embedding in images, offering new potential for data concealment and enhancing information security.

Looking ahead, after establishing a foundational framework, there are several promising avenues for future exploration. These include broadening the range of message formats to accommodate sensor and video data, refining content adaptation techniques through advanced image manipulation strategies, implementing more robust message encryption methods, exploring explainable AI in steganography for improved transparency, and considering practical applications such as digital watermarking and secure communication. By expanding the scope of covert communication methods, these advancements aim to enhance both security and adaptability.

References

1. I. Goodfellow, J. Pouget-Abadie, M. Mirza, et al., “Generative adversarial nets,” Advances in Neural Information Processing Systems, vol. 27, pp. 2672–2680, 2014.
2. S. Baluja, “Hiding images in plain sight: Deep steganography,” Advances in Neural Information Processing Systems, vol. 30, pp. 2066–2076, 2017.
3. T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” arXiv preprint arXiv:1301.3781, 2013.
4. M. A. Khan and S. Salah, “IoT security: Review, blockchain solutions, and open challenges,” Future Generation Computer Systems, vol. 82, pp. 395–411, 2018.
5. H. Shafagh, A. Hithnawi, A. R. Gervais, and S. Duquennoy, “Towards blockchain-based auditable storage and sharing of IoT data,” in Proc. ACM Workshop on Blockchain, Cryptocurrencies and Contracts (BCC’17), 2017, pp. 45–50.
6. X. Zhang, M. Zhang, and Q. Zhang, “A novel image steganography method via deep convolutional generative adversarial networks,” IEEE Access, vol. 6, pp. 38303–38314, 2018.
7. M. Hussain, M. M. Aboalsamh, and G. Muhammad, “A survey on steganography methods in cloud computing and blockchain,” IEEE Access, vol. 9, pp. 66700–66724, 2021.
8. J. Fridrich and J. Kodovsky, “Rich models for steganalysis of digital images,” IEEE Transactions on Information Forensics and Security, vol. 7, no. 3, pp. 868–882, 2012.
9. Y. Qian, J. Dong, and W. Wang, “Deep learning for steganalysis via convolutional neural networks,” in Media Watermarking, Security, and Forensics, vol. 9409, SPIE, 2015.
10. S. Sun, C. Sun, and W. Wang, “Message-oriented steganography with generative adversarial networks,” IEEE Access, vol. 8, pp. 203839–203848, 2020.
11. K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014.
12. M. Sharif and H. Abdulazeez, “A comprehensive study of generative adversarial networks and their applications,” IEEE Access, vol. 9, pp. 13958–13979, 2021.
13. A. Taspinar and T. Böhme, “A machine learning approach to detect LSB matching steganography,” in Proc. 5th ACM Workshop on Information Hiding and Multimedia Security, 2017, pp. 115–120.
14. A. Singhal and A. Bhadauria, “Review of image steganography using LSB and DCT techniques,” in Proc. IEEE Int. Conf. on Computer, Communication and Control, 2015.
15. S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
16. X. Lu, L. Wu, Z. Yuan, and J. Wu, “Double-layer GAN for coverless information hiding,” IEEE Access, vol. 8, pp. 19836–19846, 2020.
17. A. B. Hassan, H. A. Varol, and A. M. M. Ibrahim, “Performance comparison of LSB-based steganography techniques in grayscale images,” in Proc. IEEE Int. Symp. on Signal Processing and Information Technology, 2013, pp. 480–484.
18. R. Lu, X. Li, X. Liang, and X. Shen, “Secure and privacy-preserving communication in smart grids,” IEEE Transactions on Smart Grid, vol. 1, no. 1, pp. 106–114, 2010.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

19. F. Alyasiri, W. Gharibi, and M. Basamh, "A blockchain-based authentication and data protection approach for IoT healthcare systems," in Proc. IEEE Int. Conf. on Smart Applications, Communications and Networking (SmartNets), 2020.
20. N. B. Amor and A. Ben Mnaouer, "Secure and lightweight IoT blockchain architecture: Design and performance evaluation," IEEE Access, vol. 9, pp. 104728–104744, 2021.