Received Date:23rd November, 2025
Revision Date: 27th December, 2025
Accepted Date: 15th January, 2026

Image Cloak: An CNN Based Stegenography System

Aditya Gnawali[1*], Bhuwan Shrestha[2], Jyoti Bhushan Dahal[3], Mohit Raj Aryal[4]

[1]Department of Computer Engineering, Kathmandu Engineering College. E-Mail: aditya.bct77005@kecktm.edu.np
[2]Department of Computer Engineering, Kathmandu Engineering College. E-Mail: bhuwan.bct77014@kecktm.edu.np
[3]Department of Computer Engineering, Kathmandu Engineering College. E-Mail: jyoti.bct77028@kecktm.edu.np
[4]Department of Computer Engineering, Kathmandu Engineering College. E-Mail: mohit.bct77030@kecktm.edu.np

Abstract — ImageCloak is an advanced deep learning-based image steganography system that enables secure embedding and extraction of hidden data within digital images using Convolutional Neural Networks (CNNs). The system achieves high performance, with 91.48% testing accuracy in text encoding and image encoding results of SSIM: 0.8898 and PSNR: 31.12 dB, ensuring that the concealed data remains imperceptible while preserving the visual quality of the cover image. To enhance scalability and usability, ImageCloak integrates MongoDB for efficient storage and retrieval of stego-images, allowing users to manage hidden data securely over time. The system also employs robust security measures to prevent unauthorized access, ensuring that sensitive information remains protected. Additionally, it supports seamless sharing of encoded images among authorized users, making it an effective solution for private communication, secure data exchange, and long-term confidential storage. By leveraging deep learning techniques and advanced embedding methods, ImageCloak provides a reliable and scalable approach to modern steganographic applications, enhancing data privacy and security in digital communication.

Keywords — Image steganography, deep learning, Convolutional Neural Networks (CNNs), secure data embedding, image encoding, text encoding, SSIM, PSNR, MongoDB, Stego-image storage, private communication, data security, confidential information exchange.

I. Introduction

Steganography is the practice of concealing information within ordinary-looking media, such as images or audio files, so that the hidden content remains undetected. Unlike cryptography, which transforms data into unreadable formats to protect its content, steganography focuses on hiding the *existence* of the data itself. Among its various forms, image steganography is especially popular due to the widespread use of digital images and their capacity to embed information without noticeable visual changes.

Traditional methods like the Least Significant Bit (LSB) technique hide data in the smallest bits of an image's pixel values. While simple and effective for basic use, LSB-based techniques are fragile—susceptible to compression, resizing, or detection through statistical analysis. To overcome these limitations, more advanced techniques such as transform domain steganography (e.g., DCT, DWT) and AI-powered methods have been introduced. These allow data to be embedded in ways that are more resistant to tampering and harder to detect.

ImageCloak is a modern steganography project that uses Convolutional Neural Networks (CNNs) to securely hide both text and image data within digital images. Unlike traditional approaches, ImageCloak intelligently learns the best regions of an image for embedding data, preserving visual quality while maximizing security. This deep learning-based system is more robust against detection and data loss, making it ideal for secure communication, watermarking, and digital privacy in an increasingly data-driven world.

In addition to its technical advantages, ImageCloak addresses real-world needs for covert and secure data transmission. It enables reliable data extraction with minimal distortion, ensuring the integrity of the hidden information even after transmission or mild image processing. This makes it suitable for applications in fields such as journalism, defense, digital rights management, and personal privacy protection. As cyber threats grow more sophisticated, tools like ImageCloak represent the future of steganography—combining artificial intelligence with security principles to offer smarter, more resilient solutions for protecting sensitive data.

II. Problem statements

Information has become an incredibly powerful tool in the digital age. The widespread use of the Internet and advancements in communication technologies have made information easily accessible and ubiquitous. However, this vast availability of information also brings significant risks. Electronic communication is increasingly vulnerable to

* Corresponding Author

eavesdropping, hacking, and other malicious interventions, making privacy and security major concerns in today's world.

Sensitive information, such as confidential government documents, military intelligence, and personal records, can have devastating consequences if it falls into the wrong hands. The data we share or store is often at risk of being leaked, and the security of that information largely depends on the trustworthiness of the people with access to our devices.

To address this issue, there is a need for a system that can effectively conceal important information, such as images of sensitive documents, so that it can be shared securely only with trusted individuals. Such a system should be able to hide one image within another in a way that the existence of the hidden image remains undetectable to anyone except the sender and the receiver. This would ensure that sensitive data can be exchanged safely without raising suspicion or compromising privacy.

III. Objective

To develop a user-friendly web application for securely encoding and decoding images and text within a secret image while ensuring data integrity and confidentiality.

IV. Literature Review

Steganography is one of the methods used for hidden exchange of information and it can be defined as the study of invisible communication that usually deals with the ways of hiding the existence of the communicated message [1]. In this way, if successfully it is achieved, the message does not attract any attention from eavesdropper and attackers. Using steganography, information can be hidden in different embedding mediums, knows as carriers. These carriers can be images, audio files, video files and text files.

Steganography Differs from cryptography. The goal of cryptography is to secure communications by changing the data into a form that an eavesdropper cannot understand. Steganography techniques, on the other hand, tend to hide the existence of the message itself, which makes it difficult for an observer to figure out where the message is. In some cases, sending encrypted information may draw attention while invisible information will not. Accordingly, cryptography is not the best solution for secure communication; it is only part of the solution. Both sciences can be used together to better protect information. In this case, even if steganography fails, the message cannot be recovered because a cryptography technique is used as well [2].

Nearly all digital file formats, with a high degree of redundancy, are known for their being used for steganography, the redundant parts refer to those parts capable of change without any possibility to detect the alteration. Image and audio files satisfy this requirement particularly well [3]. In fact, digital images are the most used carrier file formats owing to their popularity on the internet. There are a number of steganographic techniques that enable one to hide a secret message in an image file, all of which have corresponding strong and weak points. Different steganographic techniques are used for different applications. Modern steganography categorizes two main classificatory schemes for the taxonomy of algorithms. The first distinguished algorithm is based on file type. The second is a more widely used scheme, where its categorization is based on an embedding technique.

The performance of a steganographic system can be measured using several properties. The most important property is the statistical un-detectability (imperceptibility) of the data, which shows how difficult it is to determine the existence of a hidden message. Other associated measures are the steganographic capacity, which is the maximum information that can safely embedded in a work without having statistically detectable objects [4], and robustness, which refers to how well the steganographic system resists the extraction of hidden data.

The challenge of good steganography arises because embedding a message can alter the appearance and underlying statistics of the carrier. The amount of alteration depends on two factors: first, the amount of information that is to be hidden. A common use has been to hide textual messages in images. The amount of information that is hidden is measured in bits per pixel. Often, the amount of information is set to 0.4bpp or lower. The longer the message, the larger the bits per pixel, and therefore the more the carrier is altered [5, 6]. Second, the amount of alteration depends on the carrier image itself. Hiding information in the noisy, high-frequency filled, regions of an image yields less humanly detectable perturbations than hiding in the flat regions. Work on estimating how much information a carrier image can hide can be found in [7].

New techniques have been devised in the embedding process to make the detection difficult, however it is still possible to detect the existence of the hidden message. The art and science of detection of the existence of embedded message

is called steganalysis. In addition to detection of embedded message, the main goal of steganalysis is to estimate the length of embedded message, to estimate the locations of hidden data in the stego data, to estimate the stego key used by embedding algorithm, to extract the hidden message etc. Steganalysis finds its uses in cyber forensics, cyber warfare, tracking of criminal activities over the Internet and gathering evidence for investigations in case of anti-social elements [8].

Early steganographic algorithms aimed to enhance invisibility by minimizing embedding changes in images, but this wasn't enough for security due to the strong correlations in natural images. High-dimensional features like the Spatial Rich Model (SRM) significantly improved accuracy in detecting steganography but imposed high computational costs. Few researchers have attempted to reduce the dimensionality of these features. An alternative approach incorporates selection-channel information to enhance detection, as proposed by Yang et al. and Ye et al. Despite being effective, designing such features is time-consuming, especially with the development of more secure deep learning-based steganographic frameworks. To ensure security and invisibility, some methods use object detection techniques like Faster R-CNN to select complex texture regions for embedding information. With advancements in deep learning, new steganalysis frameworks have emerged, but there remain many practical applications and opportunities for further study in real-world applications [9].

With the rise of deep learning techniques, new steganalysis frameworks have emerged. Convolutional Neural Networks (CNNs) have proven to be particularly effective for steganalysis due to their ability to learn hierarchical feature representations, making it easier to detect hidden messages [10]. Additionally, object detection methods like Faster R-CNN have been utilized to select complex texture regions for embedding, which can reduce the visibility of hidden messages in the carrier file [11]. As deep learning-based steganography continues to advance, there are numerous opportunities for improving detection techniques and developing countermeasures to protect against steganalysis [12]. Despite these advancements, ongoing research is necessary to ensure the practical applicability and security of steganography systems in real-world applications.

V. Methodology

PROCESS MODEL

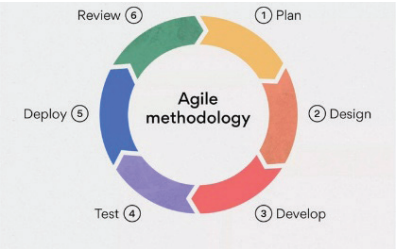

Fig 1: Agile Development Process

The Agile development process is an iterative and flexible approach to software development that emphasizes collaboration, customer feedback, and rapid delivery. It begins with planning, followed by design, development, testing, deployment, and a review phase. Each cycle, or sprint, typically lasts 1–4 weeks and results in a working product increment. After each sprint, feedback is gathered and used to improve the next iteration. This continuous cycle allows teams to adapt quickly to changing requirements and deliver high-quality software efficiently.

System Architecture

1. User Roles and Access

The system defines distinct User Roles (Admin/User) to manage access. Users can register and authenticate to access secure functionalities. Access control is implemented via JWT-based authentication, ensuring that features like encoding, decoding, and sharing are restricted based on the assigned role.

2. Frontend Development

The Frontend is built using React.js with a component-driven architecture, providing a seamless and dynamic user experience. Tailwind CSS is used for styling, ensuring a fully responsive and cross-platform compatible design. Axios handles all API interactions, and JWT manages user sessions. Core functional components include Image encoding/decoding, File upload and preview, Contact management and secure sharing, and User authentication.

3. Backend and API Design

The application utilizes a dual-backend architecture with RESTful APIs.

- The primary backend is Node.js with Express.js. It handles crucial tasks such as user management, session control, file handling, and MongoDB interactions.
- The specialized backend uses Flask (Python) to serve the deep learning model for steganography operations.

This architecture exposes major RESTful endpoints like /api/auth (for registration and login), /api/upload (for images), and/api/messages (for secure user messaging). Crucially, the /api/embed and /api/extract endpoints are primarily handled by Node.js but forward the requests to the Flask deep learning service

4. Deep Learning and Steganography Core

The core functionality is powered by a CNN-based deep learning model, trained on datasets like Fashion MNIST and ImageNet. This model supports steganography for both text and image payloads.

Inference via Flask: The Flask API provides two main endpoints: POST /api/embed, which accepts secret data and a cover image to return the stego-image, and POST /api/extract, which retrieves the hidden data from a stego-image.

- Integrity: The model uses a combination of perceptual loss and reconstruction loss during training to ensure high visual integrity of the stego-image and accurate data decoding. Images are preprocessed with normalization and resizing (224x224).

5. Data Persistence

MongoDB is used as the Database, facilitated by Mongoose for structured interactions and schema definition. MongoDB GridFS is utilized for the efficient storage and retrieval of large files, such as the stego-images. Key collections store Users (credentials, profile), Messages (text/image, sender/receiver IDs), Conversations (chat logs), and Uploads (URLs and metadata of stego-images).

6. Key Features Summary

The system provides CNN-based steganography (supporting both text and image payloads), secure communication via user authentication, real-time messaging with multimedia support, and a stego image integrity check to very data hasn't been altered during transmission.

Algorithm

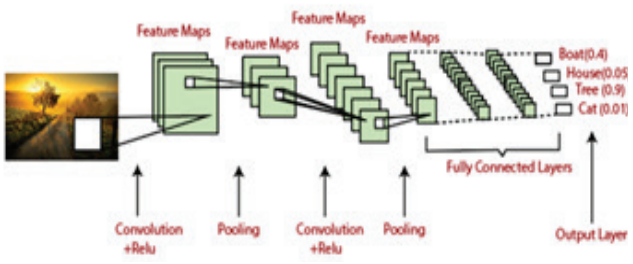

Fig 2: Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a specialized class of deep learning models designed for image recognition, classification, and processing tasks. They are inspired by the structure of the human visual system, where neurons respond to specific patterns, edges, and textures in an image. Unlike traditional machine learning models that require handcrafted features, CNNs automatically learn hierarchical features from images, making them highly efficient for computer vision applications.

CNNs consist of multiple layers that transform an input image into a meaningful representation. The core components of CNNs include convolutional layers, activation functions, pooling layers, and fully connected layers. These layers work together to extract spatial features, reduce computational complexity, and make predictions based on learned patterns.

CNN Algorithm: Step-by-Step Explanation

A CNN processes an image through multiple layers, each performing a specific task to extract relevant features and classify the image accurately. The key steps of the CNN algorithm are described below.

1. Input Layer (Image Preprocessing)

The CNN starts with an input layer that takes an image as input. The image is represented as a matrix of pixel values, where each pixel has intensity values for red, green, and blue (RGB) channels. Before feeding the image into the network, it is often resized and normalized to improve computational efficiency.

2. Convolutional Layer (Feature Extraction)

The convolutional layer is the most important component of CNNs. It applies a set of learnable filters (kernels) to the input image to extract essential features such as edges, corners, and textures. Each filter moves across the image, performing element-wise multiplication and summing up the results, producing a feature map.

Mathematical Representation:

For an image I and a filter K, the convolution operation is defined as:

$$S(i, j)=(I*K)\ (i, j)=\sum_m \sum_n I\ (m, n)K(i-m, j-n) \qquad (i)$$

where S(i,j) is the resulting feature map at position (i,j).

After applying the filters, an activation function such as ReLU (Rectified Linear Unit) is used to introduce non-linearity. ReLU replaces negative values with zero, allowing the network to learn complex patterns.

$$f(x)=\max(0,x) \quad \text{(ii)}$$

3. Pooling Layer (Down sampling)

Pooling layers reduce the spatial dimensions of feature maps while preserving the most important information. This helps in reducing computational complexity and preventing overfitting. The most common pooling operation is Max Pooling, where the maximum value from a small region (e.g., 2×2 window) is selected.

Example of Max Pooling (2×2 Window):

$$\begin{bmatrix} 1 & 3 & 2 & 1 \\ 4 & 5 & 6 & 3 \\ 2 & 8 & 7 & 4 \\ 3 & 6 & 5 & 2 \end{bmatrix}$$

After applying 2×2 max pooling:

$$\begin{bmatrix} 5 & 6 \\ 8 & 7 \end{bmatrix}$$

Pooling reduces dimensions while retaining the dominant features.

4. Additional Convolutional and Pooling Layers

CNNs often stack multiple convolutional and pooling layers to learn more abstract features. Lower layers detect simple patterns like edges, while deeper layers capture complex patterns like object shapes.

5. Fully Connected Layer (Classification)

Once the convolutional and pooling layers have extracted the features, they are passed to a fully connected layer (FC layer). The FC layer flattens the feature maps into a single vector and applies weights to determine the most probable class label.

For example, if the CNN is trained for image classification, the fully connected layer outputs probabilities for different categories (e.g., tree, cat, boat, house). The SoftMax activation function is used to convert the outputs into probabilities:

$$P(y_i) = \frac{e^{z_i}}{\sum_j e^{z_i}} \quad \text{(iii)}$$

Where z_i is the output score for class i, and the denominator ensures all probabilities sum to 1.

6. Output Layer (Prediction)

The final output layer provides the classification result based on the highest probability score. If the CNN is trained for object detection, segmentation, or steganography, the output could be a localized region, a reconstructed image, or a set of feature maps.

Data Pre-processing and Preparation

To ensure optimal model performance and smooth integration of multimedia data, several preprocessing steps were applied:

i. All images were resized to 224×224 pixels to maintain a uniform input size for the CNN model, facilitating efficient processing and consistent learning.

ii. Pixel values were normalized to a range of [0, 1] by dividing by 255. Additionally, normalization was performed separately on individual color channels to prevent any dominant color from disproportionately influencing the model's learning process.

iii. To improve dataset diversity and prevent overfitting, various transformations like random rotations, flips, zooms, and shifts were applied.

iv. The dataset was divided into:

- Training set (~90%) used for model learning.
- Test set (~10%) used for evaluating model performance.

v. Images were converted into NumPy arrays, making them compatible inputs for the deep learning model.

vi. Input text was converted into ASCII format to facilitate seamless embedding and extraction during the steganography process.

TEXT EMBEDDING AND EXTRACTION USING CNN

Step 1: Data Preparation

Cover Image Dataset:

i. The Fashion MNIST [10] dataset was used as the cover image source. It consists of 60,000 28x28 grayscale images of fashion items, such as T-shirts, dresses, and sneakers, categorized into 10 classes. This dataset was collected by Zalando Research, a subsidiary of the Zalando Company, and serves as a more complex and realistic benchmark compared to the original MNIST dataset (handwritten digits).

i. Normalization: Normalize the pixel values of the images in the range [0, 1].

Secret Text Dataset:

i. Secret text (T) was converted into a numerical

representation using ASCII encoding.

ii. Reshaping: The encoded text was reshaped into a matrix form, suitable for input to the convolutional layers of the neural network.

Step 2: Text Embedding Process (Encoder Network)

1: Preparation Network

i. Text Conversion:

The secret text was first converted into a numerical format. This involved encoding characters or words into ASCII encoding.

ii. Reshaping:

The encoded text was reshaped into a matrix form, suitable for input to the convolutional layers of the neural network.

iii. Feature Extraction:

The reshaped secret text was passed through several convolutional layers. These layers learn to extract high-level features from the text.

Different kernel sizes and activation functions (e.g., ReLU) are used to capture various patterns and semantic features of the text.

2: Hiding Network

i. Cover Image and Text Feature Concatenation:

The feature representation of the secret text (obtained from the Preparation Network) was concatenated with the cover image (C).

ii. Convolutional Processing:

The concatenated feature (cover image + encoded text) was passed through additional convolutional layers in the Hiding Network.

The network learned to modify the cover image in subtle ways, embedding the secret text without visually altering the cover image significantly.

iii. Output:

The output of the Hiding Network was the Stego Image (I'), which looks very similar to the original cover image but contains the embedded text.

Step 3: Extraction Process (Decoder Network)

i. Stego Image as Input:

The Stego Image (I') was passed through the decoder network to begin the extraction process.

ii. Feature Decoding:

The decoder network used several convolutional layers to extract the embedded features from the stego image.

iii. Text Reconstruction:

The network reconstructs the feature representation of the secret text. The feature was then decoded back into the original text using reverse ASCII encoding.

iv. Output:

The extracted text (T') was compared with the original secret text (T) to evaluate the accuracy of the embedding and extraction process.

Step 4: Model Training Process

Loss Functions:

i. *Perceptual Loss:*

Measures the perceptual similarity between the cover image and the stego image. It helped to ensure that the stego image remains visually indistinguishable from the cover image.

ii. *Reconstruction Loss:*

Measures the difference between the original secret text and the extracted secret text. It ensured that the embedded text can be accurately recovered from the stego image.

Optimization:

i. Data Splitting:

Split the dataset into training and testing sets. The training set is used to optimize the model, while the testing set is used for evaluation. The training set with 55,000 images was used to train the model, and 5,000 images in testing set were used for evaluation.

ii. Optimization:

The model was trained using an optimizer like Adam. The goal is to minimize both perceptual and reconstruction losses.

iii. Epochs:

The model was trained for several epochs to ensure the network learns to effectively embed and extract the text data.

Step 5: Evaluation Process

Evaluate Stego Image Quality:

i. Visual Evaluation:

Checked if the stego image visually resembles the cover image. This was done by comparing pixel values and using perceptual loss as an indicator of similarity.

ii. Text Extraction Accuracy:

Compared the original secret text (T) with the extracted text (T') to evaluate the reconstruction quality. This was done using metrics like accuracy.

IMAGE EMBEDDING AND EXTRACTION USING CNN

Step 1: Data Preparation

i. Load and Preprocess Cover Images:

Cover Image Dataset: We used 50,000 images from ImageNet [11] as the dataset for cover images. Each image was resized to 224×224 pixels.

Normalization: We normalized the pixel values of the cover images to the range [0,1]:

$$C' = \frac{C}{255'}, C' \in [0,1] \quad \text{.....(i)}$$

where C is the original cover image and C′ is the normalized cover image.

ii. Load and Preprocess Secret Images:

We resized the secret images to 224×224 pixels to match the cover image size.

Normalization: Similarly, the pixel values of the secret images were normalized to the range [0,1]:

$$S' = \frac{S}{255'}, S' \in [0,1] \quad \text{.....(ii)}$$

where S is the original secret image and S′ is the normalized secret image.

Step 2: Image Embedding Process (Hiding Process)

i. Prepare the Secret Image:

The secret image was passed through the Preparation Network, which applies a series of convolutional layers to extract high-level features from the secret image. Let the extracted feature map be Fs.

$$Fs = ConvNet(S') \quad \text{.....(iii)}$$

where Fs represents the feature map of the secret image after passing through the convolutional layers.

ii. Concatenate the Cover Image and Secret Image Features:

The feature map Fs of the secret image was concatenated with the normalized cover image C′. The concatenated input X is:

$$X = Concat(C', Fs) \quad \text{.....(iv)}$$

This combined representation of cover and secret image features will be used as input to the Hiding Network.

iii. Embed the Secret Image into the Cover Image:

The concatenated input X was passed through the Hiding Network to generate the Stego Image I′. This network learned to modify the cover image subtly, embedding the secret image within it. The output of this network was the stego image I″:

$$I' = HidingNetwork(X) \quad \text{.....(v)}$$

where I′ is the generated stego image.

Step 3: Secret Image Extraction Process (Decoder Network)

i. Input Stego Image:

The Stego Image I′ was passed to the Decoder Network to extract the embedded secret image.

ii. Extract Features from Stego Image:

The decoder network used convolutional layers to extract the embedded features Fs' from the stego image I′:

$$Fs' = ConvNet(I') \quad \text{.....(vi)}$$

where Fs' is the feature map extracted from the stego image.

iii. Reconstruct the Secret Image:

The extracted features Fs' were used to reconstruct the secret image S′:

$$S' = ReconstructionNet(Fs') \quad \text{.....(vii)}$$

Where S' is the reconstructed secret image.

iv. Output the Reconstructed Secret Image:

The final output was the reconstructed secret image S′, which is compared with the original secret image S to evaluate the accuracy of the extraction process.

Step 4: Model Training

i. Define Loss Functions:

Perceptual Loss: To ensure the Stego Image I′ was visually indistinguishable from the Cover Image C′, we minimize the perceptual loss:

$$Lperceptual = |\Phi(I') - \Phi(C)|_{\vee 2} \quad \text{.....(viii)}$$

where Φ is a pre-trained feature extractor (e.g., VGG) that outputs feature maps from the images.

Reconstruction Loss: To ensure the accuracy of secret image extraction, we minimized the reconstruction loss between the original secret image S and the reconstructed secret image S′:

$$Lreconstruction = \| S - S' \|_2 \quad \text{.....(ix)}$$

where $\|\cdot\|_2$ represents the L2 norm (Euclidean distance) between the original and reconstructed secret images.

ii. Data Splitting:

The dataset was split into training and testing sets. The training set with 45000 images was used to train the model, and 5000 images in testing set were used for evaluation.

iii. Model Optimization:

We used the Adam optimizer to train the model. The goal was to minimize both perceptual and reconstruction losses:

$$Ltotal = Lperceptual + \lambda Lreconstruction \quad \text{.....(x)}$$

where λ is a weight factor balancing the perceptual and reconstruction losses.

iv. Training the Model:

The model was trained for multiple epochs to learn the best parameters for both embedding and extraction.

Step 5: Evaluation

i. Evaluate Stego Image Quality:

We used Mean Squared Error (MSE) to measure the difference between the cover image C and the stego image I′:

$$MSE(C, I') = \frac{1}{N}\sum_{i=1}^{N}(Ci - Ii')^2 \quad \text{.....(xi)}$$

where N is the number of pixels, and Ci and Ii' are the pixel values of the cover and stego images.

ii. Evaluate Secret Image Extraction:

Peak Signal-to-Noise Ratio (PSNR) was used to measure the quality of the reconstructed secret image S′:

Pixel-wise Accuracy was used to measure the accuracy of the extracted secret image compared to the original secret image.

VI. System design

1. Data Collection

In ImageCloak, we utilized publicly available image datasets, specifically Fashion MNIST and ImageNet, to train our deep learning models for image steganography. The Fashion MNIST dataset was used for text embedding experiments, containing 60,000 grayscale images of clothing items. For image-to-image steganography, we leveraged 50,000 high-resolution images from the ImageNet dataset. These datasets were chosen for their diversity and suitability for training Convolutional Neural Networks (CNNs) to learn robust image features and enable secure, imperceptible data embedding within cover images.

2. Data Preprocessing

To ensure model consistency and performance, a series of essential preprocessing steps were performed on the data. All images were first subjected to resizing to a uniform dimension of 224x224 pixels to maintain consistency across the Convolutional Neural Network (CNN) pipeline. Following resizing, normalization was applied, where pixel values were scaled to the range [0, 1]; this step is critical for improving the convergence and stability of the training process. Furthermore, augmentation techniques, including flipping, rotation, zooming, and shifting, were employed to artificially increase the dataset's variability, which ultimately enhances the model's generalization capabilities and helps prevent overfitting. For the diverse data types, encoding involved converting text data into the ASCII format and reshaping it into matrices, while image data was transformed into NumPy arrays to ensure compatibility with the model architecture. Finally, the prepared datasets were subjected to splitting, with 90% designated for training and the remaining 10% reserved for testing to facilitate an accurate evaluation of the model's performance.

3. Model Selection – CNN-based Architecture

The ImageCloak system utilizes a custom Convolutional Neural Network (CNN) architecture specifically designed for steganography. The model was engineered to perform two primary tasks. First, Text Steganography involves embedding ASCII-encoded text data within Fashion MNIST images. Second, Image Steganography focuses on embedding entire images into high-resolution cover images sourced from ImageNet. Key features of this CNN model include multi-layer convolutional pipelines that are implemented for both the embedding process (acting as the hiding network) and the subsequent extraction process (acting as the revealing network). To ensure both the visual quality of the modified image and the successful recovery of the hidden data, the model incorporates both perceptual loss and reconstruction loss functions during training. Furthermore, the architecture includes custom preprocessing and preparation networks tailored for handling the distinct characteristics of both the text and image modalities.

4. Model Training and Optimization

To train the ImageCloak model efficiently and ensure scalability, we utilized the TensorFlow framework for building and training the custom CNN architecture. The model's parameters were fine-tuned across multiple epochs

using the Adam optimizer. Training relied on a combination of two key loss functions: Perceptual Loss was used to ensure that the resulting stego-image remained visually similar and indistinguishable from the original cover image, while Reconstruction Loss measured the accuracy with which the hidden data (either text or embedded image) was successfully recovered by the revealing network. The model's performance was evaluated using distinct metrics for each steganography task. For Text Steganography, the model achieved a high decoding accuracy of 91.48%. For Image Steganography, the quality of the output was assessed using Structural Similarity Index Measure (SSIM), which scored 0.8898, and Peak Signal-to-Noise Ratio (PSNR), which registered 31.12 dB, both indicating a high-quality stego-image output.

5. Backend Integration

The trained steganography model was integrated into a full-stack web application built upon the MERN stack (MongoDB, Express.js, React.js, Node.js) with Flask serving as the dedicated layer for model interaction. The Frontend, developed using React, provides an intuitive user experience allowing users to easily upload cover and secret files and view the resulting outputs. The Backend, managed by Node.js and Express.js, handles essential functions such as user authentication, file uploads, and metadata management. For core steganography operations, Flask API endpoints are responsible for model serving, handling the encoding and decoding requests by communicating directly with the CNN models. Storage for user data, generated stego-images, and message history is securely managed by MongoDB, ensuring both scalable access and data integrity. Furthermore, the application prioritizes Security: stego-images are encrypted during transmission, stored securely, and access control along with session management is strictly maintained using JWT-based authentication.

VII. Results

Model evaluation

SSIM (Structural Similarity Index Measure)

SSIM is used to evaluate the visual similarity between the cover image and the stego-image. It compares luminance, contrast, and structure to measure imperceptibility.

SSIM Score: 0.8898

Interpretation: A value close to 1 indicates high visual similarity. The stego-images produced by ImageCloak are nearly indistinguishable from the original cover images, preserving quality effectively.

PSNR (Peak Signal-to-Noise Ratio)

PSNR measures the distortion introduced during the embedding process. Higher values indicate better image quality after data embedding.

PSNR Score: 31.12 dB

Interpretation: A PSNR above 30 dB is considered excellent, confirming minimal noise and strong visual fidelity in the stego-image.

TEXT ENCODING ACCURACY

Text steganography was evaluated based on how accurately the hidden text could be extracted from the stego-image.

Training Accuracy: 95.32%

Validation Accuracy: 91.48%

Interpretation: High accuracy indicates successful encoding and decoding of secret text with minimal loss.

OBSERVATIONS AND ANALYSIS

For the ImageCloak image encoding model, the following training parameters were used:

- Epochs: 80
- Learning Rate: 0.001
- Optimizer: Adam
- Loss Functions: Perceptual Loss + Reconstruction Loss
- Framework: TensorFlow
- Dataset: ImageNet (50,000 images)
- Image Size: 224×224 pixels

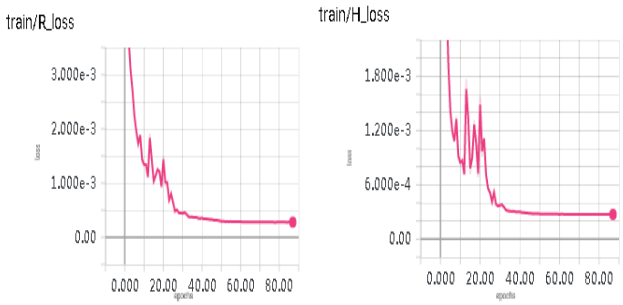

Fig 3: Training Loss for Reconstruction Loss and Hiding Loss

Table 1

Performance comparison with standard benchmark

Metric	ImageCloak Result	Ideal Range	Remarks
SSIM	0.8898	0.85-1.0	Excellent structural similarity
PSNR(db)	31.12	>30db	Low distortion, high-quality stego-images
Text Accuracy	91.48%	>90%	Reliable text extraction with low error

METHODS TO IMPROVE THE MODEL SCORE

Model performance can be improved by using high-resolution datasets like CelebA and DIV2K, applying advanced data augmentation (rotation, blur, noise), and adopting deeper CNN architecture such as ResNet, U-Net, or DenseNet. Introducing channel or spatial attention helps the model focus on the most relevant regions, while combining perceptual, SSIM, and adversarial losses enhances embedding quality. Training for more epochs with early stopping, using compressed images to simulate real-world sharing, and incorporating GAN-based steganography with a discriminator all contribute to more realistic and robust results.

Post-processing methods like denoising or sharpening can improve stego-image quality, and tuning hyperparameters – including learning rate, batch size, and layer count – further optimizes convergence and overall performance.

VIII. Conclusion

In conclusion, ImageCloak project has established a comprehensive and secure platform for deep learning-based image steganography. We have effectively integrated advanced CNN techniques for embedding and extracting hidden data, ensuring that our system not only maintains high security standards but is also user-friendly. Our architecture facilitates the long-term handling of stego-images and enables secure data exchange among authorized users. With a seamless user interface built using React.js and robust RESTful APIs for real-time image processing, we have created a solution that combines innovative deep learning methods with practical application needs.

The project also provides a scalable storage solution using MongoDB, allowing users to upload images for future use and share stego-images securely with others. Rigorous testing and validation confirm the system's reliability, making it a viable solution for secure communication, digital watermarking, and data concealment.

Overall, ImageCloak establishes a strong foundation for privacy-preserving technologies and demonstrates the potential of deep learning in steganography.

Acknowledgement

We would also like to express our gratitude to Kathmandu Engineering College for providing the necessary resources and fostering an environment that encourages academic and research endeavors. The support received from the institution has been instrumental in the successful execution of our research.

We extend our sincere gratitude to Assoc. Professor Kunjan Amatya, Supervisor for his unwavering guidance and expertise throughout our research. His insightful feedback and continuous support played a vital role in refining our work and ensuring its success. His dedication to our research has been a source of inspiration, and we are truly thankful for the valuable mentorship he provided. Our thanks are extended to Assoc. Professor Sudeep Shakya, Faculty Head of Department for his leadership and commitment to academic excellence.

Collectively, the contributions of both these individuals have shaped our research and enriched our learning journey. We are grateful for their mentorship, encouragement, and the conducive academic environment provided by Kathmandu Engineering College.

References

[1] Nagham H., Abid Y., Badlishah A., & Osamah A. (2012). Image Steganography Techniques: An Overview, 6(3), 168-187

[2] N.N El-Emam.(2007).“Hiding a large amount of data with high security using steganography algorithm.”, 3(4), 223-232.

[3] T. Morkel, J.H.P. Eloff, & M.S. Oliver. (2005). “An overview of image steganography”, 1-11.

[4] I.J. Cox, M.L. Bloom, J.A. Fridrich, and T. Kalkert. (2008). Digital watermarking and steganography, USA: Morgan Kaufman Publishers, 1-591.

[5] Fridrich J. & Goljan M. (2002). “Practical steganalysis of digital images: State of the art. In Electronic Imaging”. International Society for Optics and Photonics, 1–13.

[6] Ozer H., Avcibas I., Sankur B., & Memon M.D. (2003). Steganalysis of audio based on audio quality metrics. In Electronic Imaging. International Society for Optics and Photonics, 55–66.

[7] Yaghmaee F., & and Jamzad M. (2010). Estimating watermarking capacity in gray scale images based on image complexity. URASIP Journal on Advances in Signal Processing, 851920.

[8] Johnson N.F., & Jajodia S. (1998). Exploring steganography, seeing the unseen, 31(2), 26-34.

[9] Hussain I., Zeng J., Xinhong & Tan. A Survey on Deep Convolutional Neural Networks for Image Steganography and Steganalysis, 14(3), 1228-1247.

[10] Xu, G., Wu, H., & Shi, Y. Q. (2016). Structural design of convolutional neural networks for steganalysis. IEEE Signal Processing Letters, 23(5), 708-712.

[11] Ye, J., Ni, J., & Yi, Y. (2017). Deep learning hierarchical representations for image steganalysis. IEEE Transactions on Information Forensics and Security, 12(11), 2545-2557.

[12] Zhu, J., Kaplan, J., Johnson, J., & Fei-Fei, L. (2020). Hidden: Deep learning-based image steganography. Neural Information Processing Systems (NeurIPS).