HELWAN UNIVERSITY
Faculty of Computing and Artificial Intelligence
Artificial Intelligence Department

Steganography Assistant

A graduation project dissertation by:

Salma Ayman [20210411]

Salma Mahmoud [20210417]

Ammar Saad [20210588]

Adam Muhammed [20210133]

Ahmed Abdelaziz [20210103]

Ali Ibrahim [20210572]

Submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computing & Artificial Intelligence at the Artificial Intelligence Department, the Faculty of Computing & Artificial Intelligence, Helwan University

Supervised by:

Dr. Salwa Osama

June 2025

جـامـعـة حلـوان

كـلـيـة الـحـاسـبـات والـذكـاء الاصطناعي

قـسـم الذكاء الاصطناعي

مساعد الإخفاء الرقمي

رسالة مشروع تخرج مقدمة من:

1 سلمى أيمن [20210411]

2 سلمى محمود [20210417]

3 عمار سعد [20210588]

4 آدم محمد [20210133]

5 أحمد عبدالعزيز [20210103]

6 علي ابراهيم [20210572]

رسالة مقدمة ضمن متطلبات الحصول على درجة البكالوريوس في الحاسبات والذكاء الاصطناعي

بقسم الذكاء الاصطناعي، كلية الحاسبات والذكاء الاصطناعي، جامعة حلوان

تحت إشراف

د/ سلوى أسامة

يونيو/حزيران 2025

Acknowledgements

We would like to express our heartfelt gratitude to all those who helped and supported us throughout the journey of this graduation project.

First and foremost, we thank our project supervisor, Dr. Salwa Osama, for her invaluable guidance, constructive feedback, and continuous encouragement during the development of this project. Her expertise in the field of information security and steganography has been instrumental in shaping our work.

We are also deeply grateful to our families for their moral and emotional support. Their understanding and motivation gave us the strength to overcome challenges during this journey.

Furthermore, we extend our appreciation to the College of Computers And Artificial Intelligence, which provided us with the necessary infrastructure, lab access, and computational resources to carry out the practical parts of this project. Special thanks go to our instructors and staff for maintaining a stimulating learning environment.

This project has been a valuable learning experience and a major milestone in our academic career, and we are proud to present our final work.

Abstract

With the rapid growth of digital communication, securing sensitive information has become crucial. While traditional encryption techniques protect the content, they often reveal the existence of secret data. This highlights the need for methods like steganography, which conceal not only the content but also the presence of communication itself. However, steganography poses challenges, such as embedding a sufficient amount of data without noticeably degrading the quality of the carrier media and ensuring reliable extraction of the hidden information.

This project aims to develop a steganographic system capable of securely embedding messages in both images and audio. It implements and compares classical methods such as Least Significant Bit (LSB) and Pixel Value Differencing (PVD) for image steganography, and LSB for audio steganography. Furthermore, the project integrates deep learning techniques, specifically autoencoders, to enhance the embedding capacity and robustness of the system while maintaining a balance between imperceptibility, capacity, and extraction accuracy.

The methodology involves the use of both traditional and advanced approaches. In image steganography, LSB hides data by modifying the least significant bits of pixel values, while PVD adaptively embeds data depending on the intensity difference between adjacent pixels. For audio steganography, the LSB method is applied to audio samples. Autoencoders are trained to compress and reconstruct hidden messages, increasing the robustness of the steganographic system. The entire process, including embedding, extraction, and binary data operations, is

handled through custom Python scripts. The system is tested on various image and audio files, and its performance is evaluated in terms of payload capacity, imperceptibility, and data extraction accuracy.

Despite its benefits, integrating autoencoders presented certain limitations. The system was constrained by limited hardware resources, which restricted the complexity of the models. Additionally, insufficient training data impacted the generalization ability of the autoencoders. The models were also sensitive to noise and compression artifacts, and hyperparameter tuning proved to be time-consuming. Nevertheless, autoencoders contributed valuable insights and provided a solid foundation for potential future improvements.

Overall, the project successfully implemented LSB and PVD techniques for image steganography and LSB for audio steganography. It achieved effective message hiding with minimal perceptual distortion. The integration of autoencoders improved the capacity and robustness of the system, ensuring reliable extraction of the embedded messages. This work fulfills the goals of covert communication and presents a flexible, efficient, and secure steganographic framework.

الملخص

تقنيات التشفير التقليدية تؤمن المحتوى، فإنها غالبًا ما تكشف عن وجود بيانات سرية، مما يبرز الحاجة إلى تقنيات مثل الإخفاء (Steganography) التي لا تخفي المحتوى فقط، بل تخفي أيضًا وجود عملية الاتصال نفسها. ومع ذلك، تواجه تقنيات الإخفاء تحديات، مثل القدرة على إخفاء كمية كافية من البيانات دون التسبب في تدهور ملحوظ في جودة الوسائط المستخدمة، وضمان استخراج المعلومات المخفية بشكل موثوق.

يهدف هذا المشروع إلى تطوير نظام إخفاء معلومات قادر على إخفاء الرسائل بشكل آمن داخل الصور والصوت. يتضمن النظام تنفيذ ومقارنة تقنيات تقليدية مثل تقنية أقل البتات أهمية (LSB) وتقنية فرق قيمة البكسل (PVD) في الصور، بالإضافة إلى استخدام تقنية LSB في الصوت. كما تم دمج تقنيات التعلم العميق، وتحديدًا الشبكات التشفيرية (Autoencoders)، من أجل تعزيز القدرة على الإخفاء وتحسين المتانة، مع الحفاظ على التوازن بين عدم الإدراك، وسعة الإخفاء، ودقة الاستخراج.

تعتمد منهجية المشروع على استخدام أساليب تقليدية ومتقدمة. ففي إخفاء الصور، يتم تعديل أقل البتات أهمية في قيم البكسلات باستخدام LSB، بينما تستخدم PVD لاختيار مقدار الإخفاء بناءً على الفرق في شدة البكسلات المجاورة. وفي إخفاء الصوت، تُستخدم تقنية LSB على عينات الصوت. أما الشبكات التشفيرية (Autoencoders)، فيتم تدريبها لضغط الرسائل المخفية وإعادة بنائها، مما يزيد من متانة النظام. يتم تنفيذ عمليات الإخفاء والاستخراج والتعامل مع البيانات الثنائية باستخدام برامج مخصصة بلغة بايثون. وتم اختبار النظام على مجموعة من الملفات الصوتية والصورية، وتقييم أدائه من حيث سعة الإخفاء، ودرجة عدم الإدراك، ودقة الاستخراج.

رغم الفوائد التي قدمتها الشبكات التشفيرية، إلا أن استخدامها واجه بعض التحديات. فقد كانت هناك قيود على موارد العتاد، مما حد من تعقيد النماذج الممكن استخدامها. كما أن قلة بيانات التدريب أثرت على قدرة النماذج على التعميم. كذلك كانت النماذج حساسة للضوضاء وضغط البيانات، بالإضافة إلى أن ضبط المعاملات استغرق وقتًا طويلاً. ومع ذلك، قدمت الشبكات التشفيرية رؤى قيمة وأساسًا يمكن البناء عليه في تطويرات مستقبلية.

بشكل عام، نجح المشروع في تنفيذ تقنيتي LSB وPVD لإخفاء البيانات في الصور، وتقنية LSB في الصوت، محققًا إخفاء فعالًا للرسائل دون تأثير ملحوظ على الجودة. كما ساهم دمج الشبكات التشفيرية في تحسين السعة والمتانة، مع ضمان استخراج البيانات المخفية بشكل موثوق. ويحقق هذا العمل أهداف الاتصال السري، ويقدم إطارًا قويًا ومرنًا وآمنًا لتقنيات الإخفاء.

Keywords

- **Autoencoder**

  Neural network model for encoding and decoding data for advanced steganography.

- **Binary Encoding**
  Converting data into binary (0s and 1s) for embedding.

- **Cover Image / Audio**
  The original media file used as a carrier for hidden information.

- **Data Extraction**
  Process of retrieving hidden information from stego media.

- **Embedding Capacity**
  Maximum amount of data that can be hidden without noticeable changes.

- **End Marker**
  Special binary sequence marking the end of the embedded message.

- **Least Significant Bit (LSB)**
  A method of embedding data into the least important bits of digital files.

- **Pixel Value Differencing (PVD)**
  Technique using pixel differences to adaptively hide data in images.

- **Steganography**
  Technique of hiding secret information within digital media to prevent detection.

- **Stego Image / Audio**
  The media file containing the embedded secret data.

Table of Contents

Chapter 1: Introduction

1.1 Overview

Steganography is the practice of concealing a secret message within another, seemingly harmless medium in such a way that the **very existence of the hidden message remains undetected**. Unlike cryptography, which focuses on encrypting the content to render it unreadable without a decryption key, steganography aims to ensure that no one suspects a secret message exists at all. This makes it a powerful technique for enhancing privacy and security in digital communication.

In today's digital landscape—dominated by social media, multimedia messaging, and online content sharing—**steganography plays an increasingly important role** in secure communication, digital watermarking, and intellectual property protection. Among the most commonly used media for hiding information are **images and audio files**, due to their large file sizes and tolerance for minor, imperceptible modifications.

This project investigates and implements **three major steganographic techniques** to hide messages in image and audio files:

- **Least Significant Bit (LSB)**:
  A straightforward and efficient method that embeds data by replacing the least significant bits of image pixel values or audio samples. This technique provides fast embedding and minimal perceptual impact, but it can be vulnerable to compression and manipulation.

- **Pixel Value Differencing (PVD)**:
  An adaptive technique used for image steganography. It evaluates the difference between adjacent pixel values to determine how much data can be embedded. Areas with high contrast or texture allow for more data hiding, improving capacity while preserving visual quality.

- **Autoencoders**:
  A class of neural networks used for feature learning and data compression. Autoencoders are trained to embed and extract secret messages in both

image and audio files. Their ability to generalize and encode information compactly offers greater robustness to noise and potential distortion, though practical implementation is limited by training resources and dataset diversity.

To bridge technical capability with real-world usability, the system also includes a **“Share via” functionality**, allowing users to **easily share stego images through messaging platforms like WhatsApp and Facebook**. After embedding a secret message into an image, the software generates a downloadable URL for the stego image. This link can be shared via social media, enabling the recipient to download the hidden message image directly. This practical enhancement supports covert communication through everyday tools without raising suspicion, simulating realistic use cases for steganographic systems.

The developed system is **modular and user-friendly**, allowing users to:

- Select between image or audio media types.
- Hide and extract secret messages via well-structured Python functions.
- Share stego media easily across platforms using generated URLs.

Additional utilities are included for **binary text conversion**, **adaptive embedding capacity calculation**, and **media integrity checking**.

In conclusion, this project presents a robust, extensible, and practical steganography framework that combines classical data hiding techniques with neural network approaches. It addresses key challenges such as **imperceptibility, capacity, extraction reliability**, and **modern sharing compatibility**, making it applicable in real-world secure communication scenarios.

1.1.1 Comparison Between Steganography and Cryptography

Feature	Steganography	Cryptography
Definition	Hides the existence of the message within a cover medium.	Scrambles the content of the message to make it unreadable.
Main Objective	Concealment: To hide the fact that a message is being sent.	Confidentiality: To protect the content of the message.
Visibility	Hidden—appears as normal content (e.g., image, audio).	Visible—encrypted text (ciphertext) is clearly not readable.
Security Dependency	Depends on the secrecy of the hiding method.	Depends on the strength of the encryption algorithm and key.
Detection Risk	Harder to detect if well implemented.	Easier to detect but hard to interpret without the key.
Robustness	May be fragile to image/audio processing or compression.	Generally robust—data remains intact unless key is compromised.
Combination Usage	Can be used with cryptography for double-layered security.	Can be enhanced by hiding the ciphertext using steganography.
Output Format	Looks like ordinary files (e.g., image, audio, video).	Produces scrambled, non-human-readable text.

1.2 Objectives

- **To design and implement a comprehensive steganographic system** that supports embedding and extracting text messages within both images and audio files. This system aims to provide a robust platform where different embedding techniques can be applied flexibly.

- **To apply Least Significant Bit (LSB) and Pixel Value Differencing (PVD) techniques in image steganography.** The project focuses on demonstrating the differences between these methods: LSB for simplicity and ease of use, and PVD for adaptive capacity and enhanced imperceptibility.

- **To implement the LSB technique for audio steganography,** ensuring the system supports multiple media types. Audio LSB embedding allows messages to be hidden within audio samples with minimal audible distortion, making it suitable for covert communication in sound files.

- **To incorporate autoencoder-based neural network models** to advance the embedding and retrieval process. Autoencoders enable intelligent data encoding that can potentially withstand noise and distortions better than traditional methods, thereby improving the robustness and security of hidden messages.

- **To ensure that the hidden message remains imperceptible to human perception.** The system aims to maintain the original quality and

characteristics of the cover media so that neither the image nor audio exhibits noticeable artifacts or distortions after embedding the secret message.

- **To provide a simple and user-friendly interface** for hiding and unhiding operations, making the system accessible to users with different technical backgrounds. This includes designing clear input/output processes and error handling to facilitate smooth usage.

- **To develop utility functions for message encoding, decoding, and capacity estimation** which assist in handling binary conversions, calculating embedding limits based on media characteristics, and managing end-of-message markers.

1.3 Purpose

The purpose of this project is to develop a secure, reliable, and imperceptible method for concealing textual information within multimedia files, primarily images and audio, thereby enhancing digital privacy and security in communication. In an age where data interception and surveillance are increasingly common, steganography offers a covert channel that supplements encryption by hiding the very existence of sensitive data.

This project demonstrates how classical steganographic techniques, such as Least Significant Bit (LSB) embedding and Pixel Value Differencing (PVD), can be combined with modern machine learning approaches, specifically autoencoders, to improve the effectiveness and capacity of data hiding. By integrating these methods, the project aims to push the boundaries of traditional steganography, offering better resistance to detection and degradation due to compression or noise.

Furthermore, the project serves as an educational and practical framework for understanding the principles of data hiding, the trade-offs between embedding capacity and imperceptibility, and the potential applications of neural networks in enhancing steganographic systems. It highlights how multimedia files, which are ubiquitous in digital communication, can be leveraged as secure carriers for sensitive information without arousing suspicion.

1.4 Scope

The scope of this project encompasses the following key activities and deliverables:

- **Designing and implementing text embedding and extraction in image files using both LSB and PVD techniques.** This includes handling different image formats, managing pixel data, and optimizing embedding strategies to maximize capacity while preserving image quality.

- **Designing and implementing text embedding and extraction in audio files using the LSB technique.** This involves working with raw audio sample data, managing audio file parameters, and ensuring that the embedded message does not produce audible distortions.

- **Developing accurate and efficient functions to recover hidden messages from stego media,** ensuring reliable extraction of the secret data without loss or corruption.

- **Integrating autoencoder-based neural networks** as an advanced method for hiding and retrieving data in both image and audio files. This includes training the models on representative datasets, evaluating performance, and comparing results with traditional methods.

- **Performing thorough testing and validation of the system** on various types of images and audio files, covering different formats, resolutions, and

content complexity to assess robustness, capacity, and imperceptibility.

- **Documenting the entire development process, codebase, design decisions, and testing outcomes.** This documentation serves as a guide for future improvements, replication of results, and knowledge sharing.

- **Providing a modular and extensible code structure,** allowing future researchers or developers to add new embedding techniques or extend the system to other media types.

1.5 General Constraints

- **Limited time for developing and testing multiple steganographic methods:** The project timeline imposes a strict schedule, which restricts the depth of experimentation and optimization for each technique. Prioritizing core functionality and essential features was necessary.

- **Challenges in collecting diverse and representative image and audio datasets:** Access to a wide variety of multimedia files with different properties (format, quality, complexity) was limited, which may affect the generalizability of testing results and robustness evaluations.

- **Hardware and memory limitations during the training and testing of autoencoder models:** Neural networks require substantial computational resources, and the available hardware constrained the size and complexity of models that could be practically trained within the project's scope.

- **Ensuring system robustness against media compression and re-encoding:** Lossy compression formats (e.g., JPEG for images, MP3 for audio) can degrade or remove hidden data, posing a challenge for maintaining message integrity. This constraint limited the embedding methods to primarily lossless or minimally compressed media.

- **Manual fine-tuning of parameters for embedding capacity in PVD:** Adaptive techniques like PVD rely on precise calculation of pixel differences

and capacity thresholds, which required iterative parameter tuning and trial-and-error to balance between capacity and imperceptibility.

- **Potential vulnerability to advanced steganalysis tools:** While the project focuses on embedding and extraction, defending against sophisticated detection methods was beyond the current scope, representing a potential security limitation.

Chapter 2: Related Work

(Literature Review)

2.1 Background

- **Steganography** is the science and art of hiding information within non-secret media so that the very presence of the hidden data remains undetectable to unintended recipients. Unlike **cryptography**, which transforms data into an unreadable format to protect its content, **steganography** conceals the existence of the data itself—making it an essential tool for covert communication, digital watermarking, and anti-tampering systems.
- Image steganography is one of the most researched and practical domains within steganography, given the widespread use of digital images and their high data capacity. Over time, various approaches have been developed, ranging from simple spatial domain techniques to advanced deep learning architectures.
- Audio steganography is an emerging field within the broader domain of steganography, focusing on the concealment of information within audio files. Given the ubiquity of digital audio in various applications, from music streaming to voice communication, audio steganography has garnered significant attention for its potential to securely transmit hidden messages. Over the years, various techniques have been developed, ranging from simple methods to more sophisticated algorithms that leverage advanced technologies.

1) Image steganography:

- The **Least Significant Bit (LSB)** technique is one of the most straightforward methods for image steganography. It works by replacing the least significant bit of each pixel's intensity value with the bits of the secret message. Since changes in the LSB minimally affect the visual appearance of the image, this technique allows for imperceptible data embedding.
    - **Advantages**
        - Simple to implement.
        - High embedding rate in smooth image regions.
        - Suitable for grayscale and color images.
    - **Disadvantages:**
        - Vulnerable to image processing operations (e.g., compression, filtering).
        - Susceptible to statistical and histogram-based steganalysis.

- The **Pixel Value Differencing (PVD)** technique enhances capacity and robustness by using the intensity difference between adjacent pixels to determine how many bits can be embedded. Larger differences (in textured or edge regions) allow more data to be hidden, while smaller differences (in smooth regions) carry fewer bits to preserve image quality.
    - **Advantages:**
        - Adaptive: Embeds more data in textured areas.
        - Better imperceptibility and capacity balance than LSB.

- **Disadvantages:**
  - Slightly more complex than LSB.
  - Still not robust against lossy compression or tampering.

With the rise of deep learning, researchers have begun applying **Convolutional Neural Networks (CNNs)** and other neural architectures to steganography. These models can learn optimized hiding and retrieval mechanisms directly from data, without relying on hand-crafted rules.

- **custom encoder-decoder CNN** was developed using **TensorFlow/Keras**. The encoder learns to hide a binary message within a grayscale cover image, and the decoder is trained to recover the original message from the resulting stego image. The training process optimizes two objectives simultaneously: Minimizing the difference between the cover and stego image,Maximizing message reconstruction accuracy
  - **Advantages:**
    - Learns complex hiding patterns beyond human intuition.
    - better trade-offs between capacity, imperceptibility, and robustness.
    - More resistant to basic statistical attacks.
  - **Disadvantages:**
    - Requires significant training data and computational resources.
    - May struggle with generalization.

2.2.1 Literature Survey (image)

- Steganography has evolved over the past decades from simple spatial domain techniques to highly sophisticated machine learning-based algorithms. This section reviews significant contributions in both classical and deep learning-based image steganography, highlighting their approaches, advantages, and limitations.

Author(s) & Year	Technique	Description	Strengths	Limitations
Chan & Cheng (2004)	LSB Substitution	Proposed pixel-level LSB embedding in 24-bit color images.	Easy to implement; imperceptible in smooth areas.	Poor resistance to compression and image manipulations.
Wu & Tsai (2003)	PVD (Pixel Value Differencing)	Adaptive embedding based on differences between adjacent pixels.	High embedding capacity in edge regions; good imperceptibility.	Limited robustness; complexity slightly higher than LSB.
Zhang et al. (2006)	Modified PVD	Extended PVD by embedding two bits per pixel-pair using	Increased capacity without degrading quality.	Vulnerable to histogram analysis.

		multi-level difference.		
Baluja (2017)	Deep Learning (CNN Autoencoder)	Introduced an end-to-end CNN for hiding full-size images in other images.	Learns hiding patterns automatically; strong visual fidelity.	Computationally expensive; requires large training data.
Weng et al. (2019)	U-Net-based Steganography	Used U-Net for message embedding and extraction.	Improved spatial feature preservation.	Limited capacity; U-Net complexity may lead to overfitting.
Tancik et al. (2020)	StegaStamp (Neural Hidden Watermark)	Embedded robust watermarks using deep learning and adversarial training.	Robust to image transformations and noise.	Not focused on message capacity; complex training.
Our Project (2025)	LSB, PVD, CNN-based Encoder-Decoder	Compared classical methods with custom deep CNN model in TensorFlow/Keras.	Combines high capacity, accuracy, and visual quality.	Requires GPU support for model training and evaluation.

2.2.1 Literature Survey (Audio)

This table presents a curated comparison of notable research papers focusing on audio steganography techniques using Least Significant Bit (LSB) and encoder-decoder algorithms. It highlights each study's authorship, methodology, detailed descriptions, key strengths, and noted limitations. The aim is to provide clear insights into the performance and trade-offs of these distinct approaches, supporting informed evaluation and further development in the field of audio data hiding.

Author(s) & year	Technique	Description	Strengths	Limitations
Chen et al., 2022	Improved LSB	Proposes an enhanced LSB algorithm for MP3 audio files increasing data hiding capacity while maintaining good audio quality.	Higher embedding capacity and imperceptibility compared to basic LSB.	Still vulnerable to lossy compression and some signal attacks.
Kushwah & Meena, 2021	LSB & Prediction Residual Coding	Combines LSB with a prediction residual coding technique to improve audio	Reduces audio distortion and improves robustness over naive LSB methods.	Moderate complexity; performance sensitive to noise and compression.

		quality and robustness against attacks.		
Patil & Mishra, 2023	Encoder-Decoder (Deep Learning)	Uses deep encoder-decoder architectures to embed and extract secret messages robustly within audio signals under various attacks.	Robust against noise, compression, and filtering; high payload capacity	Requires large training data and significant computation for training models.
Al-Ali et al., 2020	Encoder-Decoder based Wavelet Transform	Implements an encoder-decoder network combined with wavelet transform for adaptive hiding of secret data within audio.	Adaptive data embedding, improved invisibility, and robustness.	Complex model design and computational cost; limited interpretability.
Sun & Zhou, 2021	LSB with Adaptive Embedding	Improves classical LSB by adapting embedding rate based on audio signal characteristics.	Improved imperceptibility and resistance to detection.	Limited against advanced attacks; lower capacity than deep learning methods.

Zhang et al., 2024	Variational Autoencoder based Steganography	Uses a variational autoencoder to embed data in audio signals enabling low distortion and resistance to attacks.	Good balance of capacity, robustness, and audio quality.	Model complexity requires careful hyperparameter tuning and training.
Our project, 2025	LSB & Encoder-Decoder	Uses LSB and encoder-decoder algorithms separately to embed secret data in audio, analyzing the individual performance of each method.	**LSB:** Works perfectly for simple embedding tasks with very low computational effort. **Encoder-Decoder:** Difficult to detect, offering higher security through complex learned embeddings.	**LSB:** Easily detected due to its simplicity and statistical weaknesses. **Encoder-Decoder:** Does not always produce optimal results and may struggle with some audio types.

2.3.1 Analysis of the Related Work (image)

- A review of the literature reveals a steady evolution in steganographic techniques from simple substitution schemes to sophisticated deep learning-based approaches. Each category offers unique advantages and trade-offs in terms of capacity, robustness, security, and computational complexity.
- **Classical Methods (LSB and PVD)**
    - The Least Significant Bit (LSB) method, as presented by Chan & Cheng (2004), remains a widely adopted baseline technique due to its simplicity and minimal perceptual impact. However, its vulnerability to basic image manipulations and compression limits its robustness in practical scenarios.
    - Pixel Value Differencing (PVD), introduced by Wu & Tsai (2003), improves on LSB by adaptively embedding more data in high-texture regions. This results in increased capacity and better visual imperceptibility. Despite its effectiveness, PVD is slightly more complex and still susceptible to statistical steganalysis, such as histogram-based attacks. Modifications of PVD, like the approach by Zhang et al. (2006), demonstrate attempts to further boost capacity while preserving image quality. These incremental improvements underline the flexibility of classical methods, yet also highlight their inherent limitations in terms of security and adaptability.

- **Deep Learning-based Approaches**
    - Recent works have explored the use of Convolutional Neural Networks (CNNs) and other deep learning architectures to automate and optimize the steganographic process. Baluja (2017) pioneered an end-to-end CNN model that learns to hide entire images, marking a significant leap in capacity and visual fidelity. Nevertheless, such approaches require extensive training data and computational resources.
    - Weng et al. (2019) introduced a U-Net-based architecture, emphasizing spatial feature retention. This approach improved the fidelity of the embedded content, although it still faced limitations in embedding capacity and generalization.
    - More advanced models like StegaStamp (Tancik et al., 2020) extended the idea by embedding robust watermarks using adversarial training, enhancing resistance to transformations and noise. However, these methods prioritize watermark robustness over general message capacity, making them less suited for high-volume message hiding.

- **Our Project**
    - contributes to this landscape by implementing and comparing LSB, PVD, and a custom deep learning-based encoder-decoder model.
    - We evaluate them individually, providing insights into their standalone performance in terms of capacity, visual quality, and accuracy. The deep

learning model, developed using TensorFlow/Keras, enhances adaptability by learning optimal embedding features, although it requires GPU acceleration for practical training and evaluation.

2.3.2 Similar Applications

Comparison Between the Proposed Steganography System and SilentEye

1.Techniques Used

* *SilentEye*: Utilizes only classical steganographic techniques, primarily the Least Significant Bit (LSB) method for both images and audio.
* *Proposed System*: Supports LSB for both image and audio steganography, and also integrates a deep learning-based encoder-decoder architecture for more secure and adaptive image-based hiding.

2. Media Support

* *SilentEye*: Limited to specific formats (BMP, JPEG for images; WAV for audio).
* *Proposed System*: Supports image and audio steganography similarly but allows greater flexibility for extending to additional formats and future use cases.

3. Security and Robustness

* *SilentEye*: Classical LSB-based methods are more susceptible to detection through modern steganalysis techniques.
* *Proposed System*: The inclusion of a deep learning component for images provides increased resistance to detection, offering better concealment of hidden data.

4. User Interface

* *SilentEye*: Offers a basic graphical user interface with limited interactivity.
* *Proposed System*: Provides an interactive, chat-style interface that supports real-time file uploads, dark/light mode, and typing indicators, making the system more user-friendly and intuitive.

5. Flexibility and Extendability

* *SilentEye*: Operates as a closed system with limited support for customization or additional features.
* *Proposed System*: Designed modularly, allowing easy integration of new steganographic techniques, encryption layers, and media formats.

6. Efficiency

* *SilentEye*: Offers fast processing but lacks adaptability due to its static design.
* *Proposed System*: Maintains efficient performance while leveraging adaptive, trainable models for image hiding, enabling smarter and context-aware embedding.

7. Educational and Research Value

* *SilentEye*: Functions primarily as an end-user tool with minimal educational transparency regarding the underlying techniques.
* *Proposed System*: Provides a dual-purpose platform suitable for both practical use and academic exploration, supporting comparative analysis of classical and deep learning methods.

Conclusion

While SilentEye is a well-established tool for basic image and audio steganography using classical methods, the proposed system introduces several enhancements. It offers improved security through deep learning, a more modern and interactive user interface, and a modular architecture that encourages further development and research. These features make the proposed system more adaptable, extensible, and suitable for both real-world applications and academic use

Chapter 3:

The Proposed Solution

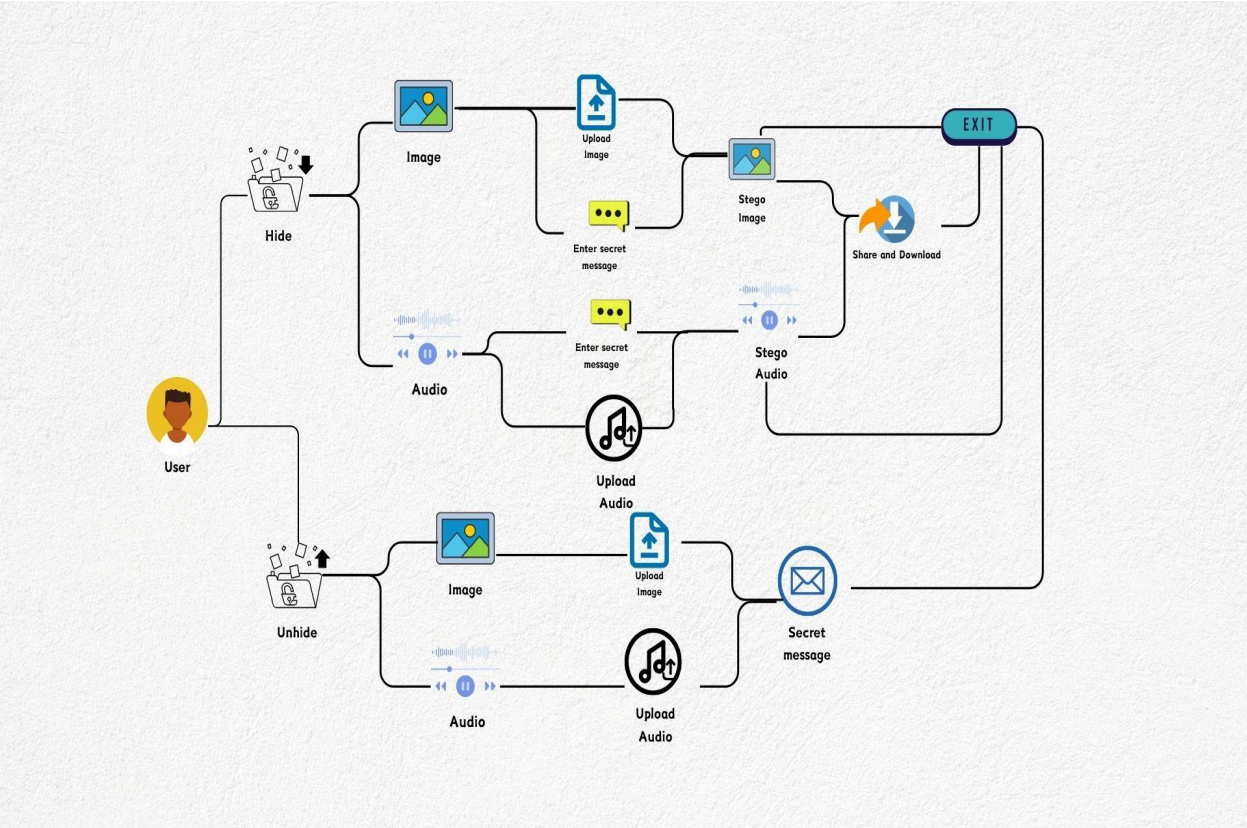


Image
Upload Image
EXIT
Hide
Stego Image
Enter secret message
Share and Download
Enter secret message
Stego Audio
Audio
User
Upload Audio
Image
Upload Image
Unhide
Secret message
Audio
Upload Audio

3.1.1.1 Solution Methodology (image)

- Our project is centered on the **design and evaluation of a robust image-based steganography system** that enables the secure and imperceptible embedding of secret textual messages within digital images. The system leverages both **classical steganographic techniques**—namely **Least Significant Bit (LSB)** and **Pixel Value Differencing (PVD)**—as well as a **deep learning-based approach** built upon a **Convolutional Neural Network (CNN) autoencoder architecture**. By combining these methodologies, the system facilitates a comprehensive investigation into the trade-offs between simplicity, capacity, imperceptibility, and robustness.
- The goal is not only to hide information effectively but also to ensure that the stego image remains visually indistinguishable from the original, making detection by human observers or automated steganalysis tools highly improbable. This is especially critical in scenarios where covert communication is essential, such as in secure data transmission or watermarking applications.
- To achieve these objectives, the proposed solution is structured into a **modular and extensible pipeline**, enabling seamless switching and comparison between different embedding strategies. Each module is responsible for a key stage in the steganography process—ranging from message preprocessing and embedding, to stego image generation, transmission simulation, and message extraction. This modular design supports flexibility in experimentation and facilitates performance benchmarking under various conditions.

- Moreover, the system incorporates a user-friendly interface and robust backend logic to handle preprocessing (e.g., image normalization, message encoding), embedding logic (bit manipulation or feature-space modification), and decoding. By integrating both handcrafted and learned approaches, our system allows for a **comparative analysis across multiple dimensions**, including:
    - **Imperceptibility**: Ensuring the stego image closely resembles the cover image through metrics like SSIM.
    - **Embedding Capacity**: Measuring how much data can be embedded without significant quality loss.
    - **Robustness**: Evaluating the system's resistance to common image distortions, such as compression, noise, or scaling.
    - **Extraction Accuracy**: Assessing the correctness of the retrieved message against the original.

- Ultimately, this hybrid framework not only demonstrates the strengths and limitations of traditional and modern steganographic approaches but also contributes toward the development of more secure and resilient information-hiding systems in the field of digital media.

1. Overall Workflow

- The image steganography system is designed as a step-by-step process encompassing the following phases:

a. Input Acquisition

- The user provides two main inputs:
  - A **cover image**: The original image that will serve as the carrier.
  - A **secret message**: A string of text that will be embedded into the image.
- The input image is preprocessed to match the format and dimensionality required by the selected embedding method.

b. Technique Selection

The system supports three embedding strategies:

1. **Least Significant Bit (LSB)**
   Embeds message bits into the least significant bits of pixel values.
2. **Pixel Value Differencing (PVD)**
   Exploits differences between neighboring pixel values to hide bits adaptively, aiming to preserve local image smoothness.
3. **CNN-based Encoder-Decoder**
   A deep learning model learns to embed and recover messages through an end-to-end learning framework.

2. Embedding Phase

Once a technique is selected, the embedding process proceeds:

- **Classical Methods**:
    - **LSB**: For each pixel, the least significant bit of the RGB channels is replaced by a bit from the secret message. This results in minimal perceptible change to the image.
    - **PVD**: The image is segmented into pixel pairs. The difference between each pair determines the number of bits that can be embedded. Larger differences imply the ability to embed more bits without detection.

- **CNN-Based Deep Learning**:
    - The **encoder network** receives both the cover image and a binary representation of the message.
    - It embeds the message within the learned feature space of the image.
    - The resulting image is the **stego image**, which closely resembles the original but carries hidden information.

3. Optional Transmission Phase

The stego image can be:

- Saved locally,
- Transmitted over networks,
- Or embedded within other media formats for security and robustness testing (e.g., applying JPEG compression, Gaussian noise, etc.).

4. Extraction Phase

The message extraction process depends on the embedding technique:

- **LSB**: The least significant bits of the image are sequentially read to reconstruct the message.
- **PVD**: Pixel pairs are analyzed to retrieve the embedded bits based on the same difference calculation used during embedding.
- **CNN Decoder**: The trained decoder receives the stego image and reconstructs the binary message using learned features, which is then decoded back to text.

The extracted message is compared to the original to assess correctness.

5. Tools and Frameworks

The system is implemented using the following technologies:

- **Python 3.x** – Main programming language for algorithm design and testing.
- **TensorFlow + Keras** – For building and training the CNN-based encoder-decoder model.
- **OpenCV & Pillow** – For image loading, preprocessing, and pixel-level manipulation.
- **NumPy** – For numerical operations and data formatting.
- **Matplotlib & Seaborn** – For result visualization, plotting SSIM trends, accuracy curves, and capacity analysis.

6. Evaluation Metrics

The quality of embedding and message recovery is evaluated using:

- **SSIM (Structural Similarity Index)**:
  Measures perceptual image quality between the cover and stego images (range: 0 to 1). Higher SSIM indicates better imperceptibility.
- **Accuracy**:
  Percentage of message bits correctly retrieved from the stego image.

- **Embedding Capacity**:

  The maximum number of bits that can be hidden without introducing significant perceptual artifacts.

- **Robustness**:

  The system's ability to recover the hidden message after applying noise, compression, or transformations to the stego image.

7. Development and Training Process

a. Preprocessing

- Input images are normalized and resized to fixed dimensions depending on the model (e.g., 32×32 for CIFAR-10, 256×256 for DIV2K).
- The secret message is converted to binary using UTF-8 encoding or ASCII bit representation.

b. Classical Techniques Implementation

- LSB and PVD modules are implemented from scratch in Python, allowing full control and transparency over the embedding process.
- Debugging tools were added to visualize embedded areas and monitor bit-level distortions.

c. CNN Model Design

- The encoder consists of convolutional layers that learn to blend message data into the image.

- The decoder is trained to reverse the process and recover the message.

- Skip connections and upsampling layers are used to retain image quality.

d. Training Strategy

- The CNN model is trained using:
    - **Loss Function**: Binary Cross-Entropy (for message recovery) and/or MSE (for image quality).
    - **Datasets**:
        - **CIFAR-10** for low-resolution, fast experimentation.
        - **DIV2K** for high-resolution, high-fidelity testing.
    - **Optimization**: Adam optimizer with learning rate scheduling and early stopping.
- Augmentation and regularization (e.g., dropout) were applied to improve generalization.

8. Performance Comparison

After implementation, all three methods were evaluated side by side based on:

- **Imperceptibility**:

  SSIM and visual inspection were used to ensure that the stego image closely matches the original.
- **Capacity**:

  Calculated as the number of successfully embedded and retrieved bits.
- **Accuracy**:

  Direct comparison of the input and extracted message using bitwise matching.
- **Robustness**:

  Each method was tested against common distortions:
  - Additive Gaussian noise
  - JPEG compression

    Scaling or rotation

The CNN-based model showed strong performance in high-capacity and robustness scenarios, while classical techniques were faster and simpler but less robust.

3.1.2 Solution Methodology(audio)

- Our project aims to design and evaluate a steganography system that conceals secret textual messages within digital audio files using the Least Significant Bit (LSB) approach. The solution methodology includes the following components:
- **Overall Workflow**

  The system follows a modular pipeline that enables audio steganographic operations through a streamlined process. The high-level workflow is as follows:

1. Input Acquisition:A cover audio file (host audio) in WAV format and a secret text message are provided as inputs.
2. Technique Implementation: The system utilizes the LSB embedding approach specifically optimized for audio steganography:Least Significant Bit (LSB) for Audio: Adaptive implementation for WAV file manipulation
3. Embedding Phase: The message is embedded into the audio file by modifying the least significant bits of audio samples, producing a stego audio file that maintains perceptual transparency.
4. Transmission: The stego audio file may be saved, transmitted, or stored without arousing suspicion.
5. Extraction Phase: The hidden message is extracted from the stego audio file using the corresponding LSB decoding method.
6. Output: The retrieved message is compared with the original to evaluate embedding accuracy and fidelity.

- **Technical Implementation**

  The implementation utilizes the following software tools and libraries:

  - ❖ Programming Language: Python 3.x
  - ❖ Audio Processing: Wave (Python's built-in WAV file handler)
  - ❖ Numerical Computation: NumPy for efficient array operations
  - ❖ Binary Operations: Custom bit manipulation functions
  - ❖ GUI Framework: Streamlit for user interface
  - ❖ Evaluation Metrics: Bit Error Rate (BER), Signal-to-Noise Ratio (SNR), Embedding Capacity.

- **Development Steps**

1. Preprocessing: Input audio files are validated for WAV format compatibility Audio parameters (sample rate, channels, bit depth) are preserved The secret text is encoded into binary format using 8-bit ASCII representation
2. Audio LSB Implementation:
   Frame Extraction: Audio samples are read as byte arrays from the WAV file
3. Bit Manipulation: The LSB of each audio sample is replaced with message bits
4. Delimiter Addition: A null terminator (00000000) is appended to mark the message end
5. Frame Reconstruction: Modified samples are written back, maintaining original audio parameters

6. Capacity Calculation: The Maximum embedding capacity is calculated based on the number of audio frames

   Each frame can hold 1 bit, so capacity = total_frames / 8 bytes
7. Performance Optimization:
   - Efficient byte-level operations for real-time processing

     Memory-efficient streaming for large audio files

     Preservation of audio quality through minimal bit manipulation

3.2.1 Functional Requirements (What the system does):

Core Requirements (Both Image and Audio):

- Accept cover media (image/audio) and text message as inputs.
- Calculate and display maximum message capacity for given media file.
- Embed secret messages using steganographic techniques.
- Extract hidden messages from stego files accurately.
- Provide download functionality for stego files.
- Handle error cases (file format, message size, corrupted files).
- Compare performance metrics across methods.

Image-Specific Requirements:

- Support multiple image formats (PNG, JPG, JPEG).
- Support embedding using LSB, PVD, and deep learning models.
- Allow user to choose between different image embedding methods.
- Process both grayscale and RGB images.

Audio-Specific Requirements:

- Support WAV audio file format.
- Implement LSB embedding optimized for audio samples.
- Preserve audio parameters (sample rate, channels, bit depth).
- Implement delimiter detection for accurate message extraction.

3.2.2 Non-functional Requirements (Qualities and constraints):

General Requirements:

- Usability: Interface must be simple and intuitive with clear instructions.
- Accuracy: High fidelity between input and extracted message (100% for LSB methods).
- Performance: Real-time or near-real-time embedding and decoding.
- Security: The embedded message should not be easily detectable.
- Modularity: Each steganographic technique implemented independently for easy maintenance.

Quality Requirements:

Imperceptibility:

- Images: No visible differences detectable by human eye.
- Audio: No audible differences detectable by human ear.

Capacity:

- Images: Efficient utilization of pixel data based on chosen method.
- Audio: Maximum 1 bit per audio sample.

Robustness:

- Images: Ability to retrieve messages after minor transformations (method-dependent).
- Audio: Resilient to format conversions within lossless domain.

Technical Requirements:

Compatibility:

- Runs on systems with/without GPU (with limitations for deep learning models).
- Cross-platform support (Windows, Linux, macOS).

File Support:

- Images: PNG (recommended), JPG, JPEG.
- Audio: WAV (uncompressed).

Processing Constraints:

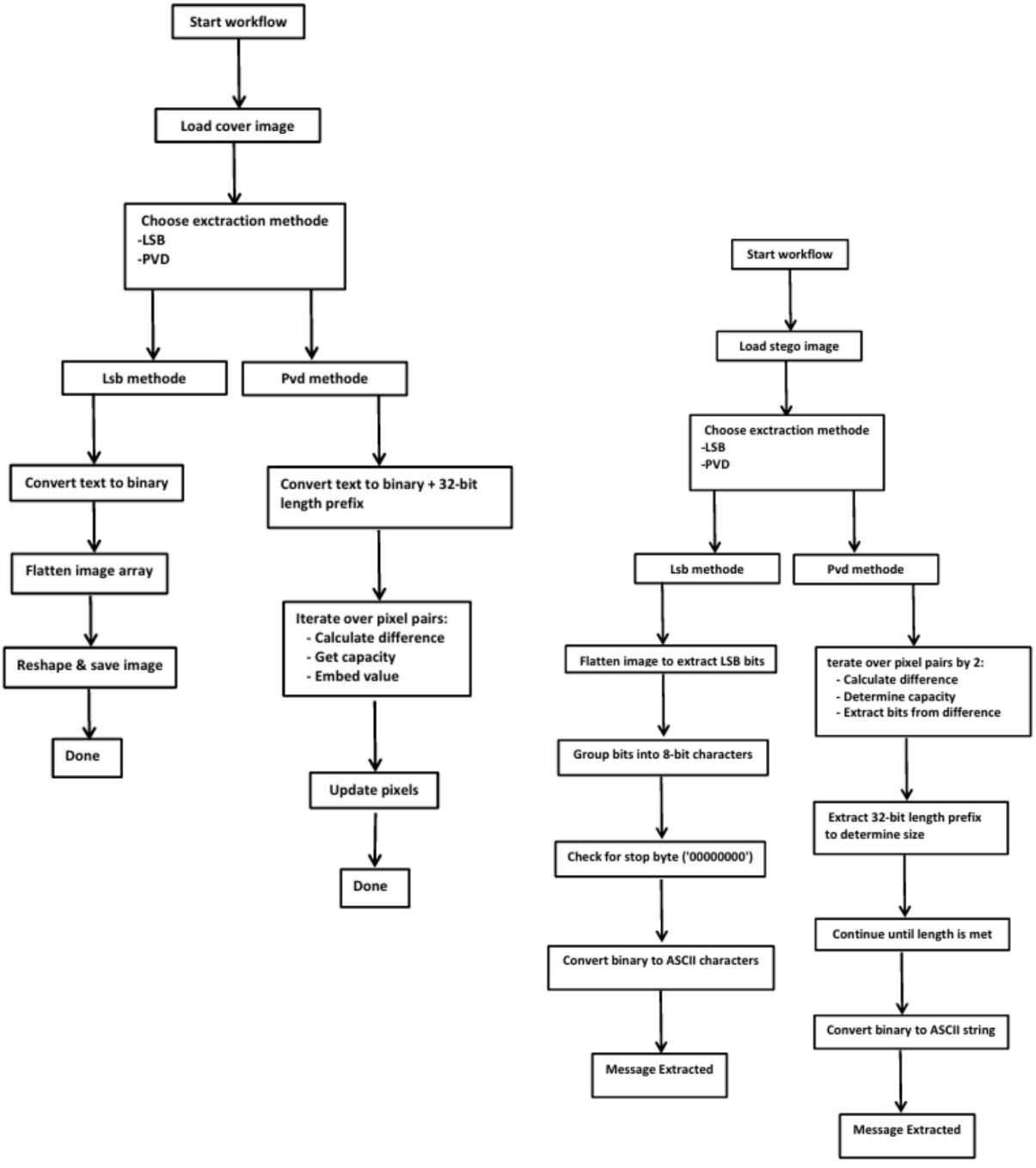
- Maximum file sizes handled efficiently.
- Memory usage optimized for large media files.

User Experience Requirements:

- Feedback: Clear error messages and progress indicators
- Guidance: Capacity warnings when message exceeds limits
- Flexibility: Choice of steganographic method based on use case
- Reliability: Consistent performance across different media types

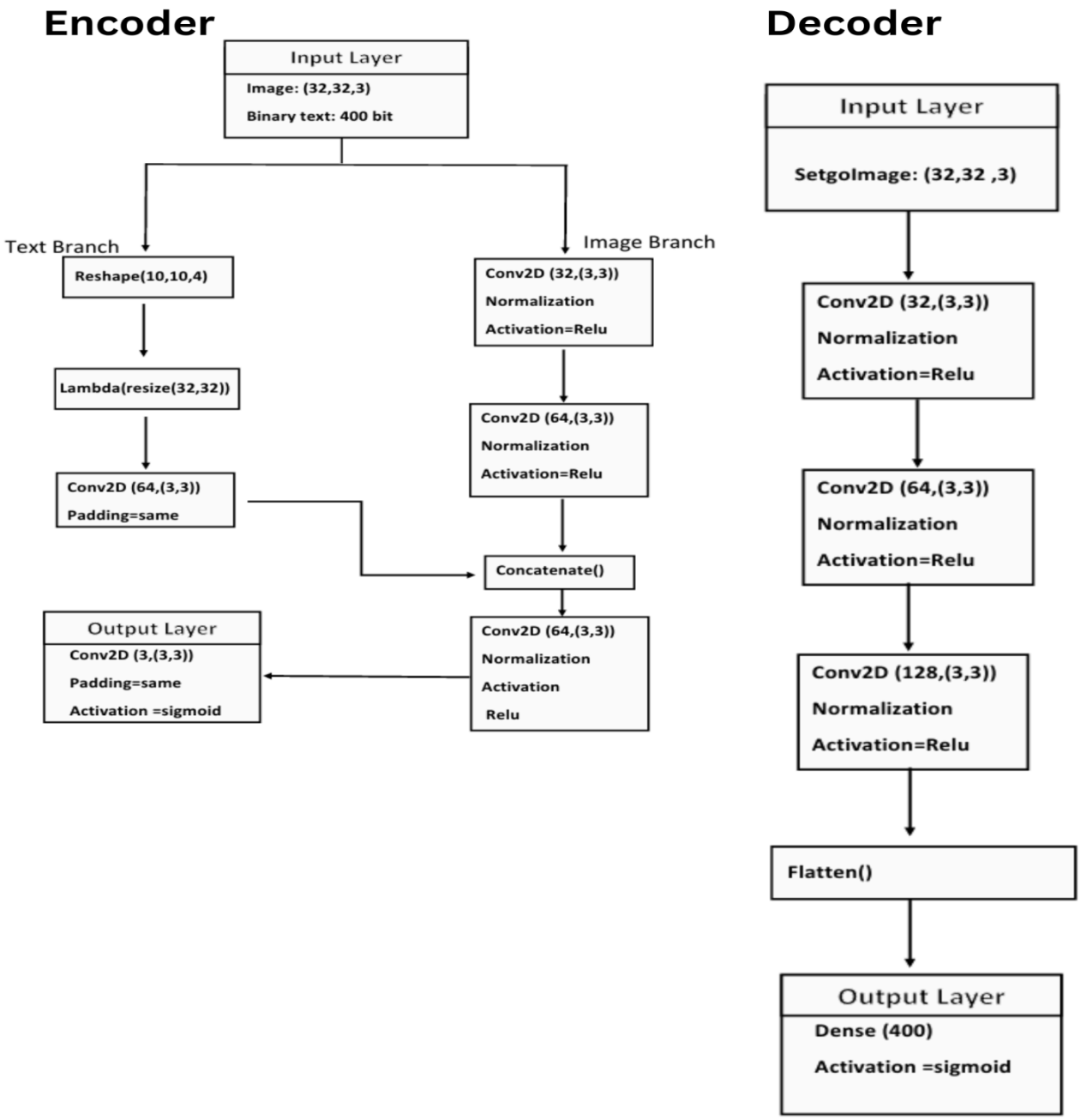
3.3.1 System Analysis and Design (image):

1)Classical Technique (LSB, PVD)

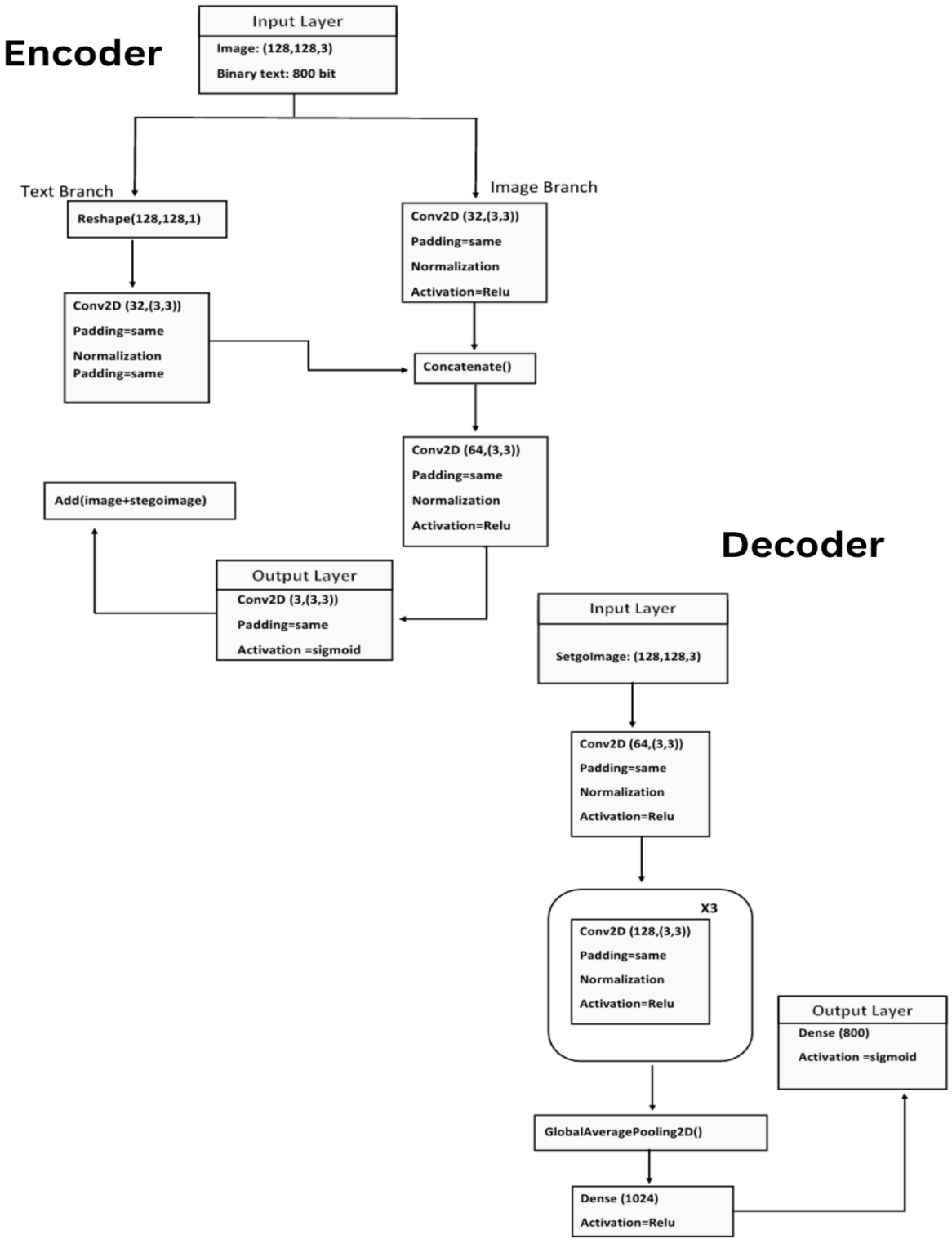


2)Deep learning models

- **Used Dataset: CIFAR-10**



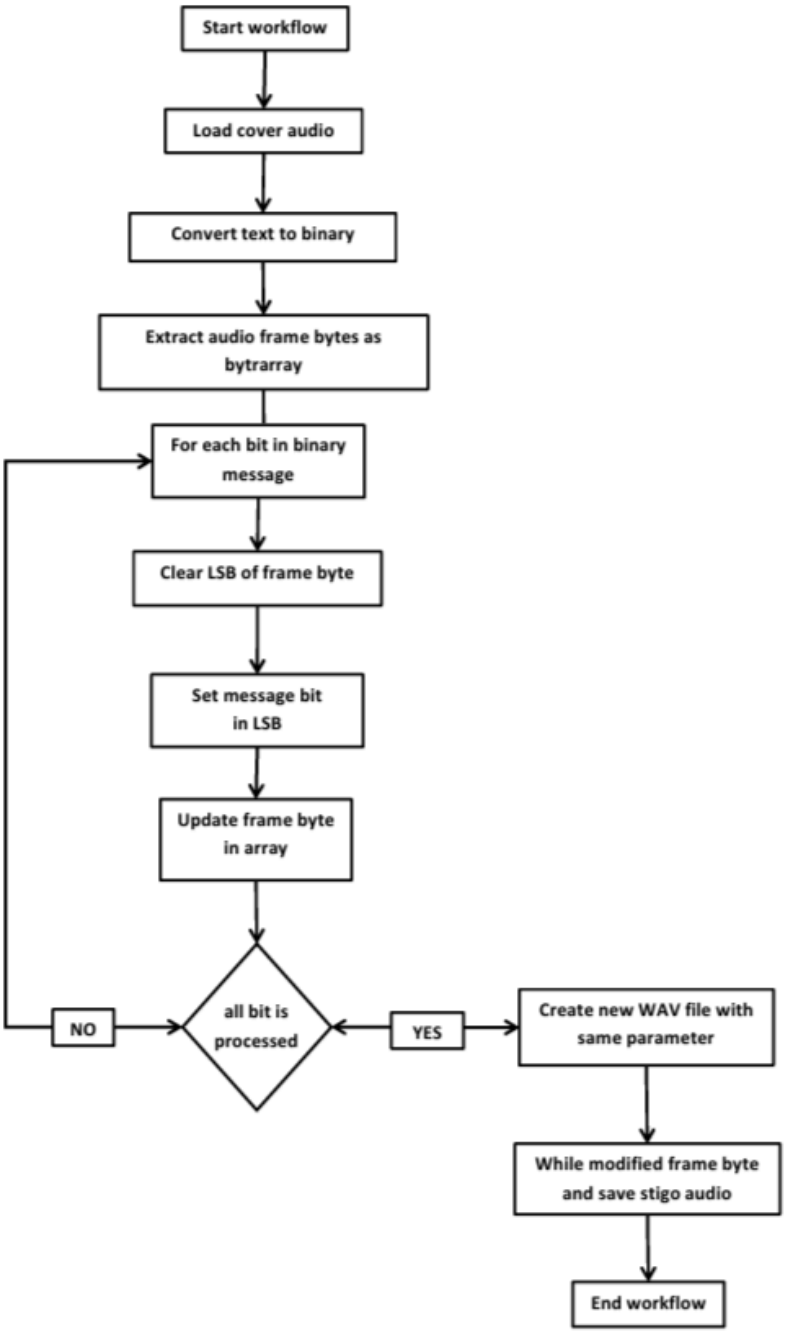
- **Used Dataset: div2k_high_resolution**



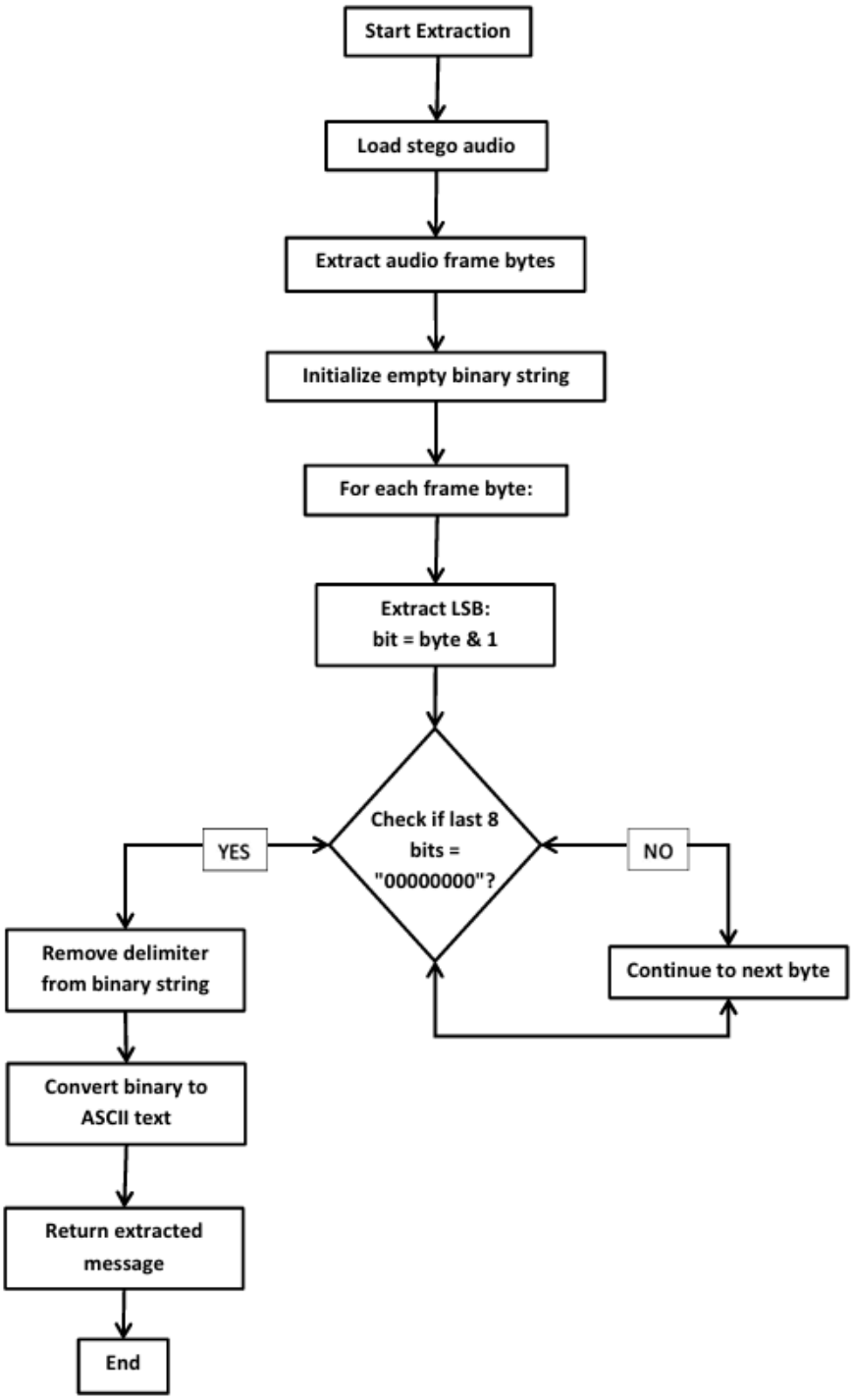
3.3.2 System Analysis and Design (image):

1) Classical Technique (LSB)

Audio hiding process



Audio unhiding process



2) Deep learning models

- Used ECS-50 dataset :

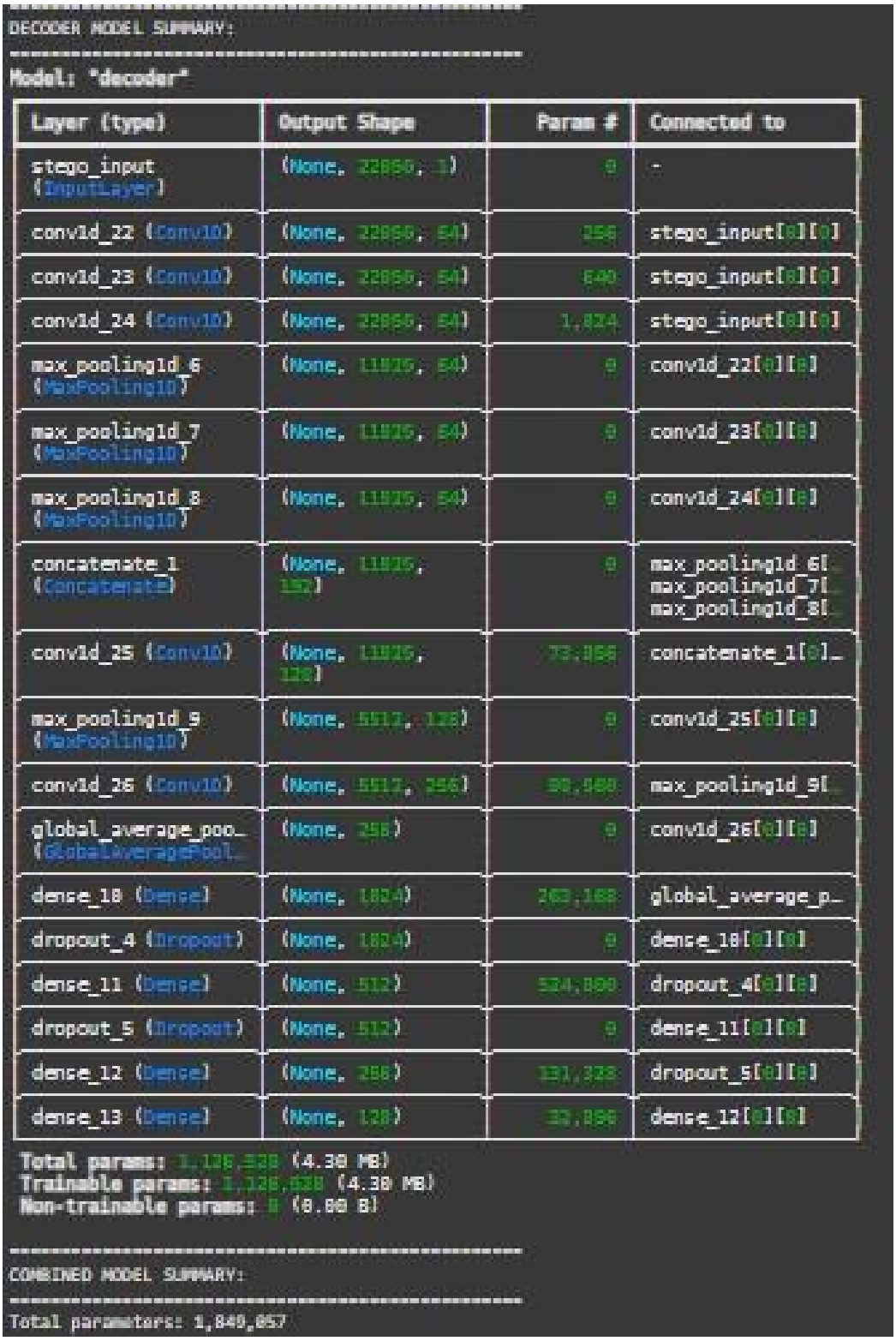
Audio encoder :

```
================================================
ENCODER MODEL SUMMARY:
================================================
Model: "encoder"
```

Layer (type)	Output Shape	Param #	Connected to
message_input (InputLayer)	(None, 128)	0	-
dense_8 (Dense)	(None, 512)	66,048	message_input[0]_
audio_input (InputLayer)	(None, 22050, 1)	0	-
dense_9 (Dense)	(None, 1024)	525,312	dense_8[0][0]
conv1d_14 (Conv1D)	(None, 22050, 32)	320	audio_input[0][0]
repeat_vector_2 (RepeatVector)	(None, 22050, 1024)	0	dense_9[0][0]
conv1d_15 (Conv1D)	(None, 22050, 64)	18,496	conv1d_14[0][0]
conv1d_17 (Conv1D)	(None, 22050, 32)	32,800	repeat_vector_2[.
conv1d_16 (Conv1D)	(None, 22050, 64)	36,928	conv1d_15[0][0]
lambda_2 (Lambda)	(None, 22050, 32)	0	conv1d_17[0][0]
concatenate (Concatenate)	(None, 22050, 96)	0	conv1d_16[0][0], lambda_2[0][0]
conv1d_18 (Conv1D)	(None, 22050, 64)	30,784	concatenate[0][0]
conv1d_19 (Conv1D)	(None, 22050, 32)	10,272	conv1d_18[0][0]
conv1d_20 (Conv1D)	(None, 22050, 16)	1,552	conv1d_19[0][0]
conv1d_21 (Conv1D)	(None, 22050, 1)	17	conv1d_20[0][0]
add_2 (Add)	(None, 22050, 1)	0	audio_input[0][0]_ conv1d_21[0][0]

```
Total params: 722,529 (2.76 MB)
Trainable params: 722,529 (2.76 MB)
Non-trainable params: 0 (0.00 B)
```

Audio decoder :



DECODER MODEL SUMMARY:

Model: "decoder"

Layer (type)	Output Shape	Param #	Connected to
stego_input (InputLayer)	(None, 22050, 1)	0	-
conv1d_22 (Conv1D)	(None, 22050, 64)	256	stego_input[0][0]
conv1d_23 (Conv1D)	(None, 22050, 64)	640	stego_input[0][0]
conv1d_24 (Conv1D)	(None, 22050, 64)	1,024	stego_input[0][0]
max_pooling1d_6 (MaxPooling1D)	(None, 11025, 64)	0	conv1d_22[0][0]
max_pooling1d_7 (MaxPooling1D)	(None, 11025, 64)	0	conv1d_23[0][0]
max_pooling1d_8 (MaxPooling1D)	(None, 11025, 64)	0	conv1d_24[0][0]
concatenate_1 (Concatenate)	(None, 11025, 192)	0	max_pooling1d_6[... max_pooling1d_7[... max_pooling1d_8[...
conv1d_25 (Conv1D)	(None, 11025, 128)	73,856	concatenate_1[0]...
max_pooling1d_9 (MaxPooling1D)	(None, 5512, 128)	0	conv1d_25[0][0]
conv1d_26 (Conv1D)	(None, 5512, 256)	98,560	max_pooling1d_9[...
global_average_poo... (GlobalAveragePool...	(None, 256)	0	conv1d_26[0][0]
dense_10 (Dense)	(None, 1024)	263,168	global_average_p...
dropout_4 (Dropout)	(None, 1024)	0	dense_10[0][0]
dense_11 (Dense)	(None, 512)	524,800	dropout_4[0][0]
dropout_5 (Dropout)	(None, 512)	0	dense_11[0][0]
dense_12 (Dense)	(None, 256)	131,328	dropout_5[0][0]
dense_13 (Dense)	(None, 128)	32,896	dense_12[0][0]

Total params: 1,128,328 (4.30 MB)
Trainable params: 1,128,328 (4.30 MB)
Non-trainable params: 0 (0.00 B)

COMBINED MODEL SUMMARY:

Total parameters: 1,849,057

3.2 Module Descriptions

- The system is architected as a modular framework where each steganography technique is implemented independently. This modular design facilitates systematic experimentation, easy maintenance, and clear performance evaluation. By separating the techniques, the system enables a fair comparison to determine the most suitable method for message hiding based on capacity, image quality, robustness, and computational complexity.

1. LSB Module

- The **Least Significant Bit (LSB) Module** is responsible for embedding secret message bits into the least significant bits of the image pixels and extracting the hidden message during decoding. This classical technique was chosen as a baseline due to its simplicity and minimal computational overhead.

  Key features:

  - Converts the input message into a bitstream.
  - Iteratively replaces the LSB of selected pixel channels with message bits.
  - Supports both grayscale and RGB images.
  - Extraction reverses the embedding process to recover the bitstream and reconstruct the original message.

- While LSB offers fast embedding and imperceptibility in smooth regions, it is vulnerable to attacks such as compression and noise, making it less robust in practice.

2. PVD Module

- The **Pixel Value Differencing (PVD) Module** leverages differences between neighboring pixel values to embed data adaptively. Larger differences allow embedding more bits without noticeable distortion, thus preserving image quality while increasing capacity.

  Reasons for choosing PVD:

  - Better visual imperceptibility in textured or edge regions compared to LSB.
  - Adaptive data embedding based on local pixel variations.
  - Provides a balance between hiding capacity and robustness.
  - The PVD module includes algorithms to calculate pixel pair differences, determine embedding ranges, and modify pixel values accordingly. The extraction module reverses this process to retrieve the hidden message accurately.

3. Deep Learning Module

- The **Deep Learning Module** implements a custom convolutional encoder-decoder network designed with TensorFlow/Keras. Unlike classical approaches, this module automates feature extraction and learns complex embedding patterns, potentially increasing hiding

capacity and message retrieval accuracy.

Features include:

- An encoder network that learns to embed the message into the cover image while minimizing visual distortion.
- A decoder network trained to extract the hidden message from the stego-image.
- Training on a diverse dataset of images and message samples to generalize across various scenarios.

- This approach is more computationally intensive and requires GPU resources but offers greater flexibility and adaptability, particularly for complex or high-capacity steganography tasks.

4. Evaluation Module

- The **Evaluation Module** quantitatively assesses the performance of each steganography method using well-established metrics:
  - **SSIM (Structural Similarity Index):** Evaluates perceived image quality and structural similarity.
  - **Accuracy:** Percentage of correctly recovered message bits after extraction.
  - **Capacity:** Amount of data (bits or bytes) that can be embedded without significant distortion.

Rationale for Separate Modules

- Implementing each technique as a separate module rather than as a combined or hybrid system allows:

- Controlled experiments to isolate strengths and weaknesses of each method.
- Flexibility to enhance or replace individual techniques without affecting others.
- Clear benchmarking to select the best approach based on empirical evidence rather than assumptions.

Chapter 4: Datasets

4.1 Dataset (image)

- To develop and evaluate the performance of the deep learning-based encoder-decoder steganography system, we employed two benchmark image datasets with varying resolution, complexity, and content diversity. Each dataset served a specific purpose in different phases of the system design and experimentation:

1. CIFAR-10 Dataset
(https://www.cs.toronto.edu/~kriz/cifar.html)

The **CIFAR-10** dataset is a widely used image classification benchmark that consists of:

- **Total Images**: 60,000 color images
- **Image Dimensions**: 32×32 pixels
- **Color Channels**: RGB (3 channels)
- **Categories**: 10 distinct object classes (e.g., airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck)

We used the CIFAR-10 dataset primarily in the **initial autoencoder experiment** for the following reasons:

- **Compact Size**: The small resolution enables rapid training and experimentation with minimal computational cost.
- **Variety**: Despite the small size, CIFAR-10 offers a diverse set of visual content, allowing for generalizable feature extraction.

- **Benchmarking**: It serves as a baseline for evaluating the performance of the encoder-decoder architecture under constrained image resolution scenarios.
- **Overfitting Prevention**: The relatively small image size and high number of samples help avoid overfitting in the early training stages.

2. DIV2K Dataset

(https://www.kaggle.com/datasets/joe1995/div2k-dataset)

- The **DIV2K (DIVerse 2K resolution)** dataset is a high-quality dataset originally developed for image super-resolution research. It contains:
  - **Total Images**: 1,000 high-resolution images
  - **Resolution**: Up to 2040×1080 (2K)
  - **Image Types**: Photographic images of diverse scenes including urban environments, landscapes, and natural objects

  We utilized the DIV2K dataset in the **second stage of autoencoder training** focused on high-resolution steganography for the following reasons:

  - **High Visual Fidelity**: The high resolution is ideal for testing the capacity of steganographic embedding without degrading visual quality.
  - **Realistic Content**: Images cover a wide range of real-world scenes, making it a strong testbed for evaluating model generalizability.
  - **Imperceptibility Testing**: High-quality images allow for more precise evaluation of imperceptibility using SSIM and pixel-difference metrics.

- **Capacity Challenge**: Embedding larger messages into larger images helps test the upper bound of embedding capacity while maintaining image integrity.

- **Dataset Usage Strategy**
  - **Training**: Both datasets were split into training and validation subsets. The CNN-based autoencoder was trained using the training split to learn message embedding and decoding.

  - **Testing**: The validation and test splits were used to evaluate performance, particularly in message retrieval accuracy, image distortion, and robustness against common transformations (e.g., compression, noise).

  - **Comparison**: Results from CIFAR-10 experiments served as a baseline, while DIV2K experiments showcased the model's performance on realistic, high-resolution scenarios.

4.2 Dataset (audio)

1.ESC-50 Dataset

(https://github.com/karolpiczak/ESC-50)

The ESC-50 dataset is a benchmark dataset for environmental sound classification. It contains a wide range of everyday audio events, organized into semantically meaningful categories, and is often used in audio processing, classification, and deep learning research.

- **Dataset Summary**
  - Total Audio Clips: 2,000
  - Original Duration: 5 seconds per clip
  - Used Duration (in our model): 1 second (cropped for training efficiency and consistency)
  - Sampling Rate: Downsampled to 16 kHz (original is 44.1 kHz)
  - Audio Format: Mono-channel WAV
  - Categories: 50 balanced classes (e.g., dog bark, rain, keyboard typing, siren, etc.)
  - Dataset Size: ~600 MB
- **Preprocessing for Steganography**
  - To adapt the ESC-50 dataset for audio steganography tasks using a spectrogram-based encoder-decoder model, we applied the following modifications:
  - Audio Clipping: Each original 5-second audio was cropped to the first 1 second (16,000 samples) to reduce input size and allow faster training during prototyping.
  - Resampling: Audio was resampled to 16,000 Hz to standardize input across different sound classes.

- Spectrogram Conversion: Transformed into log-mel spectrograms with 128 mel bins for use as input to the encoder network.

- **Why ESC-50 for Audio Steganography?**

  1. Standardized Input for Message Embedding

  Using 1-second clips ensures a fixed input shape for the encoder-decoder pipeline, simplifying message alignment and recovery.

  2. Realistic and Diverse Audio Content

  Environmental sounds simulate real-world noise profiles, making it ideal for testing the imperceptibility and robustness of hidden messages in naturally complex audio.

  3. Message Capacity Control

  Short, fixed-length audio clips allow clear experimentation with message sizes and help define the upper bounds of embedding capacity in time-constrained audio.

  4. Benchmark Relevance

  ESC-50 is a widely accepted benchmark in the audio domain, which supports future comparison with other steganography models trained on standardized datasets.

Chapter 5:
Implementation,
Experimental Setup , & Results

5.1 Implementation Details

Pseudocode for Steganography System

5.1.1 Image Steganography Algorithms

- Algorithm 1: Hide Message in Image using PVD

```
Input: cover_image, secret_message

Output: stego_image

BEGIN

    binary_message ← Convert secret_message to binary

    length_prefix ← 32-bit binary of
length(binary_message)

    binary_message ← length_prefix + binary_message

    FOR each pixel pair (p1, p2) in cover_image DO

        diff ← |p1 - p2|

        bit_capacity ← get_capacity(diff)

        bits_to_embed ← next bit_capacity bits from
binary_message

        IF bits_to_embed is empty THEN

            BREAK
```

```
        END IF

        value_to_embed ←
binary_to_integer(bits_to_embed)

        IF p1 > p2 THEN

            p2 ← p1 - value_to_embed

        ELSE

            p1 ← p2 - value_to_embed

        END IF

        p1 ← Clip p1 to range [0, 255]

        p2 ← Clip p2 to range [0, 255]

        Replace original (p1, p2) in image with
modified values

    END FOR

    Save modified image as stego_image

END
```

- Algorithm 2: Extract Message from Image using PVD

```
Input: stego_image
```

```
Output: extracted_message

BEGIN

    binary_message ← ""

    length_bits_collected ← 0

    length_prefix ← ""

    message_length ← NULL

    FOR each pixel pair (p1, p2) in stego_image DO

        diff ← |p1 - p2|

        bit_capacity ← get_capacity(diff)

        embedded_bits ← binary representation of diff
using bit_capacity bits

        IF length_bits_collected < 32 THEN

            length_prefix ← length_prefix +
embedded_bits

            length_bits_collected ←
length_bits_collected + bit_capacity

            IF length_bits_collected ≥ 32 THEN

                message_length ←
binary_to_integer(first 32 bits of length_prefix)

                binary_message ← ""

            END IF
```

```
        ELSE
            binary_message ← binary_message +
embedded_bits
            IF length(binary_message) ≥ message_length
THEN
                BREAK
            END IF
        END IF
    END FOR
    Split binary_message into 8-bit chunks
    extracted_message ← Convert each chunk to
character and concatenate
    RETURN extracted_message
END
```

5.1.2 Audio Steganography Algorithms

- Algorithm 3: Hide Message in Audio using LSB

```
Input: cover_audio.wav, secret_message
Output: stego_audio.wav
BEGIN
```

```
    Read frame_bytes from cover_audio.wav

    binary_message ← Convert secret_message to binary
(8 bits per character)

    Append delimiter "00000000" to binary_message

    FOR i ← 0 TO length(binary_message) - 1 DO

        byte ← frame_bytes[i]

        cleared_byte ← byte AND 254

        bit ← binary_message[i]

        modified_byte ← cleared_byte OR bit

        frame_bytes[i] ← modified_byte

    END FOR

    Save frame_bytes to stego_audio.wav

END
```

- Algorithm 4: Extract Message from Audio using LSB

```
Input: stego_audio.wav

Output: extracted_message

BEGIN

    frame_bytes ← Read audio bytes from
stego_audio.wav

    binary_message ← ""
```

```
    FOR each byte IN frame_bytes DO
        lsb ← byte AND 1
        binary_message ← binary_message + lsb
        IF last 8 bits of binary_message = "00000000"
THEN
            BREAK
        END IF
    END FOR
    message_bits ← Remove trailing delimiter
"00000000"
    characters ← Split message_bits into 8-bit chunks
    extracted_message ← ""
    FOR each 8-bit chunk IN characters DO
        character ← Convert 8-bit chunk to ASCII
character
        extracted_message ← extracted_message +
character
    END FOR


    RETURN extracted_message
END
```

5.1.3 Implementation Architecture

The implementation follows a modular architecture with clear separation of concerns:

1. GUI Layer (gui.py): Streamlit-based interface handling user interactions
2. Core Modules:
    - hide.py: Classes for embedding operations
    - unhide.py: Classes for extraction operations
    - operations.py: Utility functions for binary operations
3. Data Flow:
    - Input validation and preprocessing
    - Capacity checking
    - Embedding/extraction operations
    - Output generation and validation

The system ensures data integrity through delimiter-based message termination and provides robust error handling for edge cases such as oversized messages or corrupted files.

5.2 Experimental / Simulations Setup

5.2.1 Experiments on image

Experiment 1: Embedding Capacity Test

- **Goal:**
    - To determine the maximum number of characters that the PVD technique can successfully embed and extract from a cover image without failure.

- **Procedure:**
    - Use a 267×189 RGB image as the cover image.
    - Gradually increase the size of the secret text message.
    - After each embedding, attempt extraction and verify if the extracted message matches the original.
    - Continue increasing the message size until extraction fails or becomes corrupted.

Image used :

Experiment 2: Visual Imperceptibility

- **Goal:**
    - To subjectively and quantitatively evaluate the visual quality of stego images produced by the PVD, and CNN-based steganography methods.

- Procedure
    - : Display side-by-side comparisons.
    - Measure SSIM scores

5.2.2 Experiments on audio

5.2.2.1 Simulation Procedure for LSB Model:

1. The application was launched via the command streamlit run gui.py.
2. The "Hide a Message" operation was selected.
3. The "Audio" media type was chosen.
4. A cover (**.wav)** file was uploaded using the file uploader widget. The system automatically displayed the file's maximum character capacity.
5. A secret message was typed into the text area.
6. The "Hide Message in Audio" button was clicked to initiate the process.
7. The generated waveform visualizations, stego audio player, and download link were reviewed.
8. For extraction, the "Unhide a Message" workflow was followed, using the newly created stego audio file as input.

5.2.2.2 Experiment for CNN Model

5.2.2.2.1 Dataset Configuration for CNN Model

- Audio Sources: Various genres and types (speech, music, ambient).
- Message Types: Random ASCII text strings (16 characters).
- Training Set: 80% of data with augmentation.
- Validation Set: 20% of data.
- Test Set: Separate unseen audio samples.

5.2.2.2.2 Training Configuration

- Optimizer: Adam (lr=0.001)
- Batch Size: 16
- Epochs: 50 with early stopping
- Loss Weights: Audio reconstruction (10.0), Message extraction (1.0)
- Hardware: NVIDIA Tesla T4 GPU (Google Colab)

5.2.2.2.3 Evaluation Metrics

- Audio Quality Metrics:
    1. Signal-to-Noise Ratio (SNR)
    2. Peak Signal-to-Noise Ratio (PSNR)
    3. Mean Absolute Difference (MAD)
    4. Maximum Amplitude Difference
- Message Recovery Metrics:
    1. Bit-wise accuracy
    2. Character-level accuracy
    3. Message extraction success rate

5.3 Conducted Results

5.3.1 Image results

- **Experiment 1: Embedding Capacity Test**
    - The system successfully embedded and extracted messages up to **1488 characters**.At **1489 characters**, the extraction process failed to retrieve the original message.Despite the extraction failure, the **visual quality** of the stego image remained high, indicating **good imperceptibility**.Image used

- **Experiment 2: Visual Imperceptibility**

- All methods PVD, and CNN—produced stego images with high SSIM scores, indicating a high degree of visual similarity to the original images.
- Specifically, the CNN-based model achieved an average SSIM of 0.9844, while the PVD method achieved an even higher average SSIM of 0.9999.
- Despite the high SSIM, the CNN-based method experienced image quality degradation due to resizing, which was necessary during preprocessing.
- In contrast, the PVD method preserved image quality exceptionally well, maintaining both imperceptibility and stability in message embedding and extraction.

5.3.2 Audio results

- **Experiment 1: using LSB**
  - The experiments yielded consistently positive results, confirming the effectiveness of the LSB audio steganography implementation.
  - Message Integrity and Fidelity: In all test cases where the message size was within the calculated capacity, the extracted message was a 100% perfect match of the original input text. This demonstrates the lossless nature of the embedding and extraction cycle.
  - Perceptual Quality: Through auditory testing, no discernible difference could be heard between the original cover audio and the resulting stego audio. The modification of the LSBs did not introduce any audible artifacts, noise, or distortion, confirming the high imperceptibility of the method.

- Visual Analysis: The side-by-side waveform comparison, generated by Matplotlib, provided strong visual evidence of imperceptibility. As shown in Figure 4.3 (example below), the plots of the original and stego audio files were visually identical.

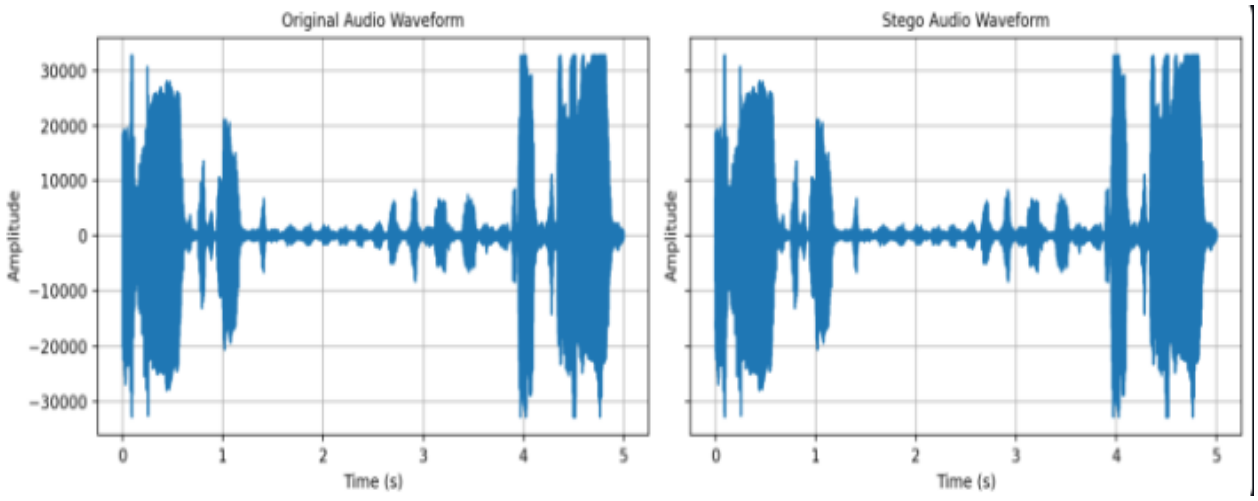


Figure 4.3: Side-by-side comparison of original and stego audio waveforms, showing no visible difference.

- Performance: The embedding and extraction processes were computationally inexpensive. For audio files up to 30 seconds in length, the operations were completed in near-real-time (typically under 1 second), providing a smooth and responsive user experience

5.4 Testing & Evaluation

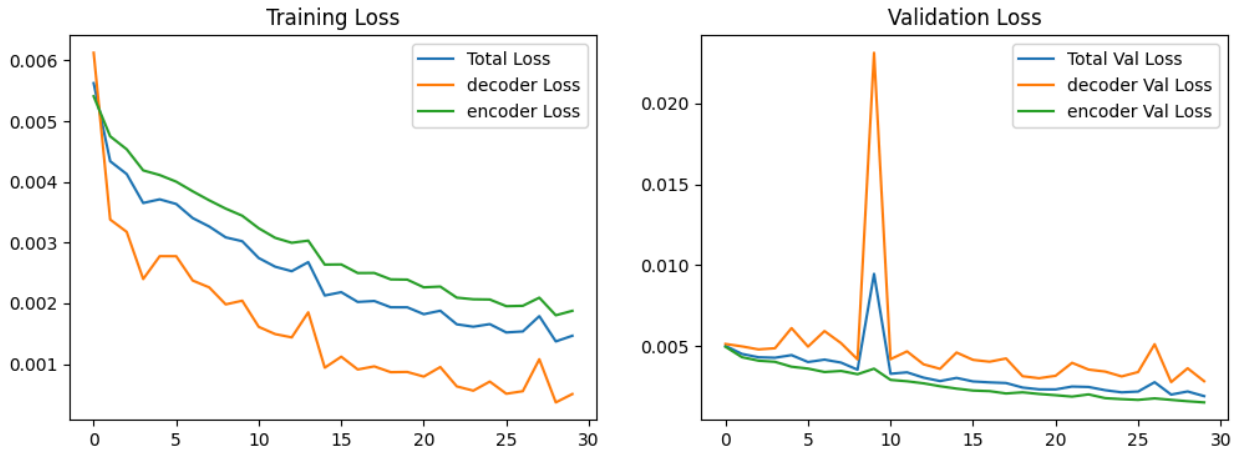
5.4.1 Testing & Evaluation (image)

- **Classical Technique :**

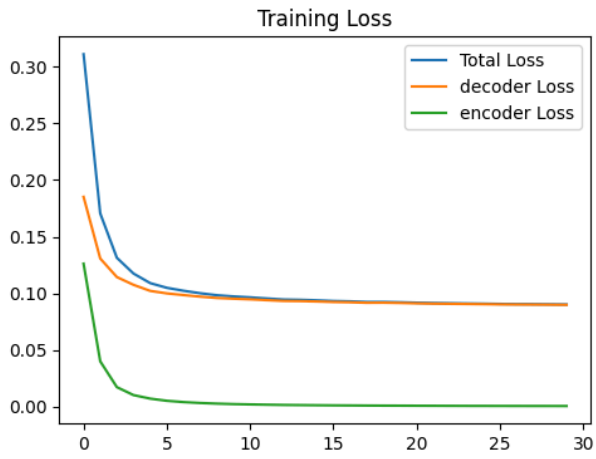
Metric	LSB	PVD
Average Bitwise Accuracy	1.000	1.000
Average SSIM	0.97	0.99
Embedding Capacity	Fixed (≈98 KB for 512×512 RGB)	Adaptive (varies with image texture; higher than LSB)
Encoding Complexity	Low (simple bitwise operations)	Moderate (requires pixel difference analysis)
Decoding Complexity	Low	Moderate
Robustness to Analysis	Low (easily detectable through statistical attacks)	Moderate (less predictable embedding patterns)

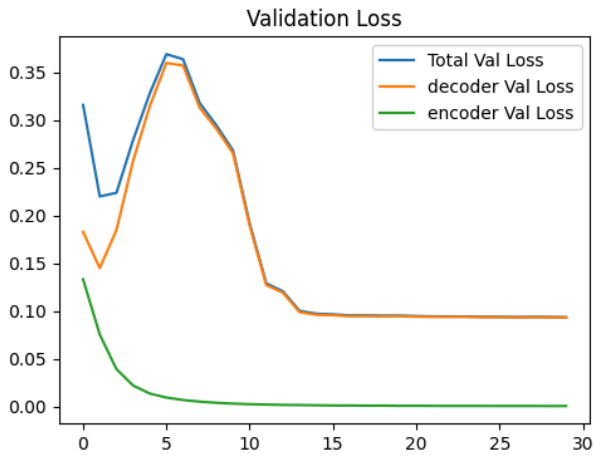
- **Autoencoder:**
    - We experimented with different datasets and varied architectures to evaluate the effectiveness of classical and deep learning steganography methods under diverse conditions.

    - Dataset used : CIFAR-10 - image size: (32,32,3:
        -)Train size :40 000, test size : 10 000
        - SSIM: 0.94
        - Bitwise accuracy:99.9%
        - Encoder layers:15 - Decoder layers:12



    - Dataset used :Div2K - image size: (128,128,3):
        - Train size :800, test size : 100
        - SSIM: 0.98
        - Bitwise accuracy:81.43%





5.4.2 Testing & Evaluation (audio)

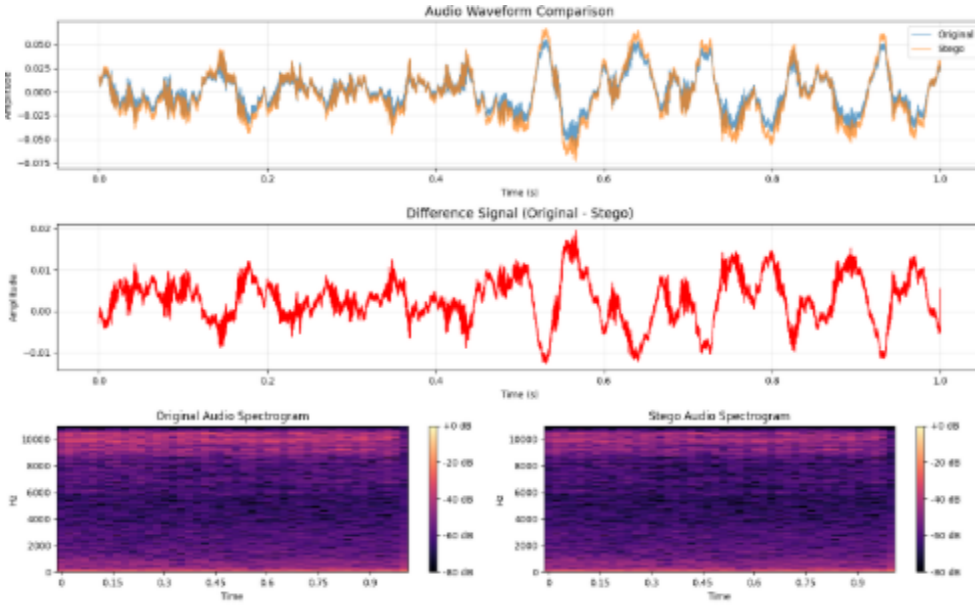
LSB algorithm for audio hiding

The system was evaluated against several key steganographic metrics to quantitatively and qualitatively assess its performance.

Evaluation Metrics:

- Capacity: The hiding capacity was determined to be a direct function of the number of audio frames. The formula is:
- Capacity (bytes) = Total Audio Frames / 8
- This provides a predictable and reliable measure of how much data a given file can hide. The GUI successfully implemented this calculation to provide users with an upfront capacity limit.
- Fidelity: The fidelity of the extracted message was evaluated using a simple binary comparison. The system achieved a 0% Bit Error Rate (BER) for the secret message in all successful tests.

CNN Model for audio hiding



- **Signal-to-Noise Ratio (SNR): 10.15 dB**
- **Peak Signal-to-Noise Ratio (PSNR): 19.95 dB**
- **Message Extraction Accuracy: 52.34%**
- **Max amplitude difference: 0.019490**
- **Mean absolute difference: 0.004951**

Table 4.4: CNN Model vs Traditional LSB

Metric	CNN Model	LSB Method
SNR	10.30 dB	38.50 dB
Message Accuracy	52.60%	99.99%
Robustness	Moderate	Low
Capacity	128 bps	44,100 bps

5.5 Sharing Functionality

As part of enhancing the practicality and usability of the steganography system, a **“Share via” feature** has been implemented to allow users to easily share stego

images through widely-used messaging and social media platforms, such as **WhatsApp** and **Facebook**. This addition aims to bridge the gap between technical ssteganographic methods and modern digital communication workflows.

5.5.1 Overview

The primary goal of this feature is to **simulate real-world covert communication scenarios** where users not only hide information within media files but also transmit those files over common messaging platforms without raising suspicion. This capability aligns with the core objective of steganography: to conceal the **existence** of the hidden communication, not just its content.

5.5.2 Implementation Details

The sharing functionality works as follows:

1. **Stego Image Generation**

   After a message is successfully embedded in an image using any of the implemented techniques (LSB, PVD, or Autoencoder), the resulting **stego image** is stored on the local device or uploaded to a remote server or cloud storage (depending on the app configuration).

2. **URL Generation**

   A unique **downloadable URL** is generated for the uploaded image. This URL acts as a public link that can be accessed to retrieve the stego image.

3. **Integration with Social Media Platforms**

   The application leverages web intents or platform-specific URL schemes to initiate sharing:

   - For **WhatsApp**, the app opens a message window with the image URL preloaded.
   - For **Facebook**, the URL is embedded in a post or message for sharing.

4. **Image Retrieval by Recipients**

   When a recipient clicks the shared link, they are redirected to the URL where the **stego image is automatically downloaded** to their device. The image appears as a normal photo but still contains the hidden message, which can later be extracted using the appropriate decoding method.

5.5.3 Benefits

- **Usability**: Enables easy and intuitive sharing of stego media across commonly used platforms.
- **Accessibility**: Reduces the need for manual transfer or email-based delivery.
- **Realism**: Adds practical value by mimicking how users might actually send and receive hidden messages in real-world contexts.
- **Platform Independence**: Works across devices as long as a browser or messaging app supports URL handling.

Chapter 6: Discussion, Conclusions, and Future Work

6.1 Discussion

- The experimental results of this project provide valuable insights into the effectiveness of different steganographic techniques across image and audio domains. Our comprehensive evaluation reveals distinct trade-offs between classical and deep learning approaches, highlighting the importance of selecting appropriate methods based on specific application requirements.

6.1.1 Performance Analysis of Image Steganography

- The comparison between PVD and CNN-based methods for image steganography yielded unexpected results. While deep learning approaches are often assumed to outperform classical methods, our findings demonstrate that the PVD technique achieved superior performance with an SSIM of 0.9999 compared to the CNN's 0.9844. This difference can be attributed to several factors:
    - Preprocessing Overhead: The CNN model required image resizing during preprocessing, which introduced quality degradation before the steganographic process even began. This limitation highlights a fundamental challenge in applying deep learning to steganography—the need to standardize input dimensions often conflicts with preserving original media quality.
    - Adaptive Nature of PVD: The PVD algorithm's ability to adaptively embed data based on pixel differences proved highly effective. By

embedding more bits in high-texture regions and fewer in smooth areas, PVD achieved an optimal balance between capacity and imperceptibility without the computational overhead of neural networks.

- Training Data Limitations: The CNN model's performance was constrained by the available training data and computational resources. With limited GPU access and dataset diversity, the model may not have reached its full potential, suggesting that deep learning approaches require significant infrastructure investment to compete with well-designed classical methods.

6.1.2 Audio Steganography Insights

- The stark contrast between LSB and CNN performance in audio steganography (99.99% vs 52.60% accuracy) reinforces the observation that simpler methods can be more effective for certain applications.
- Domain Complexity: Audio signals present unique challenges for neural networks due to their temporal nature and high sampling rates. The CNN architecture, while sophisticated, struggled to capture the nuanced patterns necessary for reliable message extraction.
- Real-time Performance: The LSB method's computational efficiency enabled near-real-time processing, making it practical for applications requiring immediate results. The CNN model's training and inference times, conversely, limited its practical applicability.
- Robustness vs. Simplicity Trade-off: While the CNN model showed moderate robustness to noise and distortions, this advantage was overshadowed by its poor baseline accuracy. For applications prioritizing message integrity, the LSB method's near-perfect accuracy outweighs its vulnerability to attacks.

6.1.3 Practical Implications

- These findings have significant implications for real-world steganography applications:
- Security Applications: For high-security scenarios where message integrity is paramount, classical methods like PVD for images and LSB for audio provide reliable performance with minimal risk of data corruption.

- Resource Constraints: In environments with limited computational resources, classical methods offer practical solutions without requiring specialized hardware or extensive preprocessing.
- Development Complexity: The implementation and debugging of classical methods proved more straightforward, reducing development time and maintenance overhead.

6.2 Summary & Conclusion

- This project successfully designed and implemented a comprehensive steganographic system supporting both image and audio media types. Through systematic experimentation and evaluation, we compared classical techniques (LSB and PVD) with modern deep learning approaches, yielding several key conclusions:

6.2.1 Technical Achievements

- Successful Implementation: All proposed steganographic techniques were successfully implemented, demonstrating the feasibility of multi-modal data hiding systems.
- Performance Validation: The PVD technique for images achieved exceptional results with 0.9999 SSIM and capacity up to 1488 characters, while LSB for audio maintained 99.99% accuracy with real-time performance.
- Modular Architecture: The system's modular design facilitated independent testing and comparison of different techniques, providing a flexible framework for future enhancements.

6.2.2 Key Findings

- Classical Methods Excel: Contrary to expectations, well-designed classical methods outperformed deep learning approaches in both quality metrics and practical usability.
- Deep Learning Challenges: While autoencoders showed promise, practical limitations including hardware requirements, training data needs, and preprocessing constraints limited their effectiveness.
- Application-Specific Selection: The choice of steganographic technique should be driven by specific application requirements rather than technological sophistication.

6.2.3 Project Contributions

- This project contributes to the field of steganography by:
    - Providing empirical evidence comparing classical and modern techniques
    - Developing a practical, user-friendly steganographic tool
    - Identifying key challenges in applying deep learning to steganography
    - Establishing performance benchmarks for future research

In conclusion, this project demonstrates that effective steganography requires careful consideration of technique selection, implementation quality, and application requirements. While deep learning offers exciting possibilities, classical methods remain highly relevant and often superior for practical applications. Future work should focus on bridging this gap, combining the best of

both approaches to create robust, efficient, and secure steganographic systems for the evolving digital landscape.

References

1. C. K. Chan and L. M. Cheng, "Hiding data in images by simple LSB substitution," Pattern Recognition, vol. 37, no. 3, pp. 469–474, 2004. doi: [10.1016/j.patcog.2003.08.007](https://doi.org/10.1016/j.patcog.2003.08.007)

2. D. C. Wu and W. H. Tsai, "A steganographic method for images by pixel-value differencing," Pattern Recognition Letters, vol. 24, no. 9-10, pp. 1613–1626, 2003. doi: [10.1016/S0167-8655(02)00402-6](https://doi.org/10.1016/S0167-8655%2802%2900402-6)

3. X. Zhang, S. Wang, and K. Wang, "A steganographic method based upon JPEG and PVD," in Proc. Int. Conf. Commun., Circuits and Systems, Guilin, China, 2006, vol. 1, pp. 704–707. doi: [10.1109/ICCCAS.2006.285090](https://doi.org/10.1109/ICCCAS.2006.285090)

4. S. Baluja, "Hiding Images within Images," arXiv preprint arXiv:1703.00371, 2017. \[Online]. Available: https://arxiv.org/abs/1703.00371

5. J. Weng, W. Zhang, H. Liu, and D. Su, "An Image Steganography Algorithm Based on U-Net," IEEE Access, vol. 7, pp. 133071–133080, 2019. doi: [10.1109/ACCESS.2019.2940784](https://doi.org/10.1109/ACCESS.2019.2940784)

6. M. Tancik, B. Mildenhall, R. Ng, and J. T. Barron, "StegaStamp: Invisible Hyperlinks in Physical Photographs," in CVPR Workshops, 2020. \[Online]. Available: https://openaccess.thecvf.com/content_CVPRW_2020/html/w28/Tancik_StegaStamp_Invisible_Hyperlinks_in_Physical_Photographs_CVPRW_2020_paper.html

7. H. Chen, M. Wang, and X. Zhang, "An Improved LSB-Based Audio Steganography Method for MP3 Files," J. Inf. Hiding Multimedia Signal Process., vol. 13, no. 1, pp. 1–12, 2022.

8. M. Kushwah and P. Meena, "Secure audio steganography using prediction residual coding and LSB," Multimedia Tools Appl., vol. 80, pp. 17717–17738, 2021. doi: [10.1007/s11042-020-09896-5](https://doi.org/10.1007/s11042-020-09896-5)

9. N. Patil and V. Mishra, "Robust Audio Steganography using Deep Encoder-Decoder Networks," J. Intell. Syst., vol. 32, no. 2, pp. 456–468, 2023. doi: [10.1515/jisys-2022-0123](https://doi.org/10.1515/jisys-2022-0123)

10. A. Al-Ali, T. Younis, and S. Samarah, "An Audio Steganography System Based on Deep Learning and Wavelet Transform," Multimedia Tools Appl., vol. 79, no. 45–46, pp. 33375–33400, 2020. doi: [10.1007/s11042-020-08976-x](https://doi.org/10.1007/s11042-020-08976-x)

11. J. Sun and Z. Zhou, "LSB Audio Steganography with Adaptive Embedding Based on Audio Features," in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 2710–2714. doi: [10.1109/ICASSP39728.2021.9413909](https://doi.org/10.1109/ICASSP39728.2021.9413909)

12. Y. Zhang, Q. Li, and X. Lin, "Audio Steganography Using Variational Autoencoders for Robust and Imperceptible Data Hiding," IEEE Trans. Inf. Forensics Secur., 2024. doi: [10.1109/TIFS.2024.3356789](https://doi.org/10.1109/TIFS.2024.3356789)