IVAN FRANKO NATIONAL UNIVERSITY OF LVIV

INTERMATHS

UNIVERSITY OF L'AQUILA

Double-Degree Master's Programme "InterMaths"

Applied and Interdisciplinary Mathematics

Master of Science	Master of Science
Applied Mathematics	Mathematical Engineering
IVAN FRANKO NATIONAL UNIVERSITY OF LVIV	UNIVERSITY OF L'AQUILA

Master's Thesis

Steganalysis using deep neural networks

Supervisor

Assoc. Prof. Yuriy Muzychuk

Candidate

Viktor Seredovych
Student ID (UAQ): 279260
Student ID (LVIV): 27210477C

ACADEMIC YEAR 2022/2023

Abstract

In this article, we discuss the deep learning steganalysis approach for detecting images with secretly embedded content. The entire research was divided into two main parts: preparing the steganalysis dataset and training a deep-learning classification model for recognizing steganographic objects. We focus on well-known content-based steganography algorithms such as J-Uniward, UERD, J-MiPOD, and LSB. We explored various deep learning architectures and techniques to achieve high detection accuracy for some of the methods like LSB and UERD. Conversely, the J-Uniward and J-MiPOD algorithms proved to be resilient to steganalysis attacks at low embedding rates.

Keywords: machine learning, deep learning, steganalysis, steganography, image processing

Contents

Chapter 1

Introduction

As our world relies heavily on secure data storage and processing, cryptography is commonly used to encrypt data. Traditional cryptography operates under the assumption that an eavesdropper knows about the secret communication and might attempt to reveal it. However, there are situations where it is essential to conceal the very existence of the communication, and that is where steganography becomes an essential tool.

Steganography is a practice that aims to hide information by disguising the existence of a secret communication itself. It has various applications, including confidential communication, secret data storage, and protecting data from tampering through invisible digital watermarks in images.

However, steganography can also be misused for malicious purposes. Malware developers can embed harmful code within innocent-looking files, allowing them to bypass common threat detection tools and security analysts. This makes it crucial to develop quick and reliable algorithms to detect dangerous steganographic files, which is the main object of this article's research.

Currently, most of the existing steganalysis methods approaches involve deep learning techniques, since they have proven to be highly effective for both concealing and detecting steganographic data. Deep learning steganalysis methods outperform statistical techniques due to their capacity for automatic feature learning and adaptability. While statistical methods depend on predefined features, deep learning models extract relevant features directly from data, leading to increased accuracy and robustness. This enables deep learning-based steganalysis to effectively detect concealed content across a variety of media types and steganographic methods. That is why, in this research, we focus on the deep neural network approach for detecting hidden steganographic data.

1.1 Terminology

This section provides an overview of key concepts in steganography that are essential for understanding how steganography works and its potential applications.

Definition 1.1 *Steganography is the technique of hiding secret data within an ordinary, non-secret file or message to avoid detection.*

Definition 1.2 *Steganography files, which are also referred to as carriers, are files that have been embedded with hidden information as a result of steganography use.*

Definition 1.3 *Covers are files that can potentially be used as carriers. That could be any file as long as there exists an embedding method that supports it.*

Definition 1.4 *Clean files are files that are untouched and do not have any information embedded with steganography.*

Definition 1.5 *Steganalysis is the reverse process of steganography and could be described as a process of detecting steganography by looking at variances between bit patterns and unusually large file sizes.*

Definition 1.6 *The embedding rate refers to the ratio between the size of a payload and its cover file. For example, if a cover image is 10 MB in size carrying*

1 MB of hidden data, the embedding rate of the image is 0.1. *In the context of JPEG images, the embedding rate is measured with the bpnzAC metric. It identifies the density of hidden information within an image, particularly in the non-zero AC coefficients of the JPEG-compressed image. A higher bpnzAC value implies more hidden information is embedded within the image, while a lower value indicates less embedded information.*

1.2 Problem statement

A steganographic system could be defined as a mechanism that embeds secret message $m \in \mathcal{M}$ in a cover object $x \in \mathcal{C}$ using a secret shared stego key $k \in \mathcal{K}$, obtaining the steganographic file $y \in \mathcal{C}$ that carries m [1].

The set $\mathcal{M}$ is the set of all possible messages, $\mathcal{K}$ is the set of all stego keys, and $\mathcal{C}$ is the set of all available cover objects. The embedding mechanism could be described using the embedding mapping

$$Emb : \mathcal{C} \times \mathcal{M} \times \mathcal{K} \to \mathcal{C}, y = \mathrm{Emb}(x, m, k)$$

This mapping has an inverse extraction mapping that extracts the hidden message from the steganographic file

$$\mathrm{Ext} : \mathcal{C} \times \mathcal{K} \to \mathcal{M}, \mathrm{Ext}(y, k) = m.$$

Given that x and y are vectors, the places where embedding changes occur, $x_i \neq y_i$, can be chosen randomly or using some specific selection rule such as the content-adaptive rule. For example, if the selection rule uses information calculated from some local neighborhood, it is considered to be a content-adaptive selection rule.

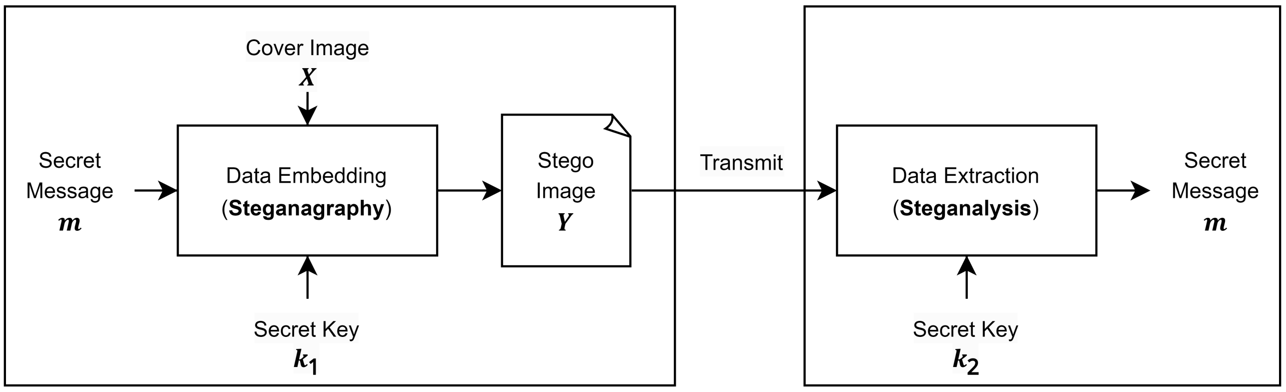


Figure 1.1: Example of encoding-decoding flow: The secret message is encoded into the cover image by using some steganography algorithm and a secret key, forming the stego image. This image is transmitted to the receiver where the secret message is extracted by using inverse mapping. Alternatively, the message could be extracted by a steganalysis attack.

In this study, the cover objects are images, while the embedding messages consist of textual data.

Chapter 2

Literature review

This study is heavily influenced by the research of Mikołaj Płachta et al. ”Detection of Image Steganography Using Deep Learning and Ensemble Classifiers” (2022) [2]. In their paper, the authors discuss the problem of detecting steganographically embedded JPEG images by employing deep neural networks. Their research aims to compare the robustness of the most popular steganography techniques and provide insights into the effectiveness of using shallow and deep learning algorithms.

Another study dedicated to JPEG-based steganalysis was conducted by Fridrich et al. at 2007, called ”Statistically Undetectable Jpeg Steganography: Dead Ends Challenges, and Opportunities” [1]. The authors explore the steganographic capacity of JPEG images by measuring the largest payload that can be undetectably embedded with respect to the best steganalysis methods available at the moment of writing.

In the paper by Cogranne et al. (2019), the authors present the ALASKA challenge, a steganalysis competition designed to emulate the conditions faced by forensic steganalysts. It highlights key differences from the 2010 BOSS challenge such as examining a wide variety of image sources and employing steganographic schemes suitable for color JPEGs. The authors detail the challenge's core components, including the RAW image dataset, cover image generation methods, and embedding scheme specifics. They present initial results, analyze the influence of various parameters, and conclude by discussing both strengths and weaknesses, as well as future directions for practical steganalysis challenges.

Holub et al. in the research "Universal distortion function for steganography in an arbitrary domain" (2014) [3] introduce the universal wavelet relative distortion (UNIWARD) method for steganography in an arbitrary domain such as the JPEG domain (J-Uniward). The proposed method minimizes a suitably defined distortion function to make steganographic content more difficult to detect. The authors demonstrate that their method matches or outperforms other state-of-the-art methods in the spatial and JPEG domains.

Guo et al. (2015) introduced the UERD method in the study "Using Statistical Image Model for JPEG Steganography: Uniform Embedding Revisited". The authors revisit uniform embedding for JPEG steganography using a statistical image model. The authors propose a new framework that incorporates a statistical model to achieve better performance in terms of secure embedding capacity for steganalysis. Their findings suggest that the proposed method can effectively reduce the detectability of steganographic content.

Cogranne et al. (2020) [4] present a novel method for steganography in JPEG-compressed images, referred to as J-MiPOD, based on minimizing the detection accuracy using a Gaussian model of independent DCT coefficients. The authors also explore the challenges of embedding in color JPEG images, noting that more research is needed to better understand how to deal with color channels in JPEG images.

Chapter 3

Cover images

Steganography can be utilized with a variety of digital image formats. However, JPEG, PNG, and BMP are the predominant selections. JPEG which is a prevalent format for photographic images, is employed by 78.0% of all websites, as indicated by [5]. This percentage is just slightly lower than the most popular format, PNG, which is utilized by 82.1% of websites. However, the JPEG format is particularly appealing for steganography applications due to its lossy compression algorithm and noise resilience. JPEG compression uses the Discrete Cosine Transform (DCT) and quantization techniques to integrate hidden data while minimizing the impact on visual quality. It is challenging to identify the existence of steganographically disguised information due to the inherent noise in JPEG pictures. These are the main reasons why in this study we are mostly focused on JPEG images as carriers of steganographically embedded data.

3.1 JPEG format

The JPEG (Joint Photographic Experts Group) format is a popular standard for compressing digital images. It is a lossy compression format, meaning some data is lost during the compression process, but the resulting file size is much smaller than the original uncompressed image. Understanding the underlying principles of JPEG compression is essential for grasping the principles of associated steganography algorithms. The JPEG compression algorithm is divided into the following steps:

- **Color space conversion:** JPEG often uses the YCbCr color space instead of the typical RGB color space. YCbCr separates the image into luminance (Y) and chrominance (Cb and Cr) components. This step does not reduce the amount of data because it only changes how the same information is represented.
- **Subsampling:** The chrominance components (Cb and Cr) are often subsampled, meaning their resolution is reduced compared to the luminance component (Y). This step exploits the fact that the human eye is less sensitive to chrominance (color) than luminance (brightness) detail.
- **Block division:** The image is divided into 8x8 pixel blocks. If the image size is not a multiple of 8, it is padded to make it so.
- **Discrete Cosine Transform (DCT):** Each 8x8 block undergoes a transformation called the Discrete Cosine Transform. The DCT converts the spatial-domain pixel values into a frequency-domain representation. This results in a set of 64 coefficients, where the top-left coefficient (DC) represents the average color of the block, and the other coefficients (AC) represent different frequencies of color variation within the block. At this step, there is still no data reduction occurs.
- **Quantization:** This step is where the lossy compression occurs. The DCT coefficients are divided by a quantization matrix, which contains values representing the human visual system's sensitivity to different spatial frequencies. The coefficients are then rounded to integers. Higher quantization values result in more aggressive compression and more loss of detail.

- **Reordering and run-length encoding:** The quantized coefficients are reordered in a zigzag pattern, grouping similar frequencies together. This ordering helps increase the efficiency of the next step: run-length encoding.

- **Run-length encoding:** is a lossless compression technique that replaces sequences of the same value with a single value and a count.

- **Entropy coding:** The final step is to apply entropy coding, usually through Huffman coding or arithmetic coding. This step further compresses the data by assigning shorter binary codes to more frequently occurring patterns.

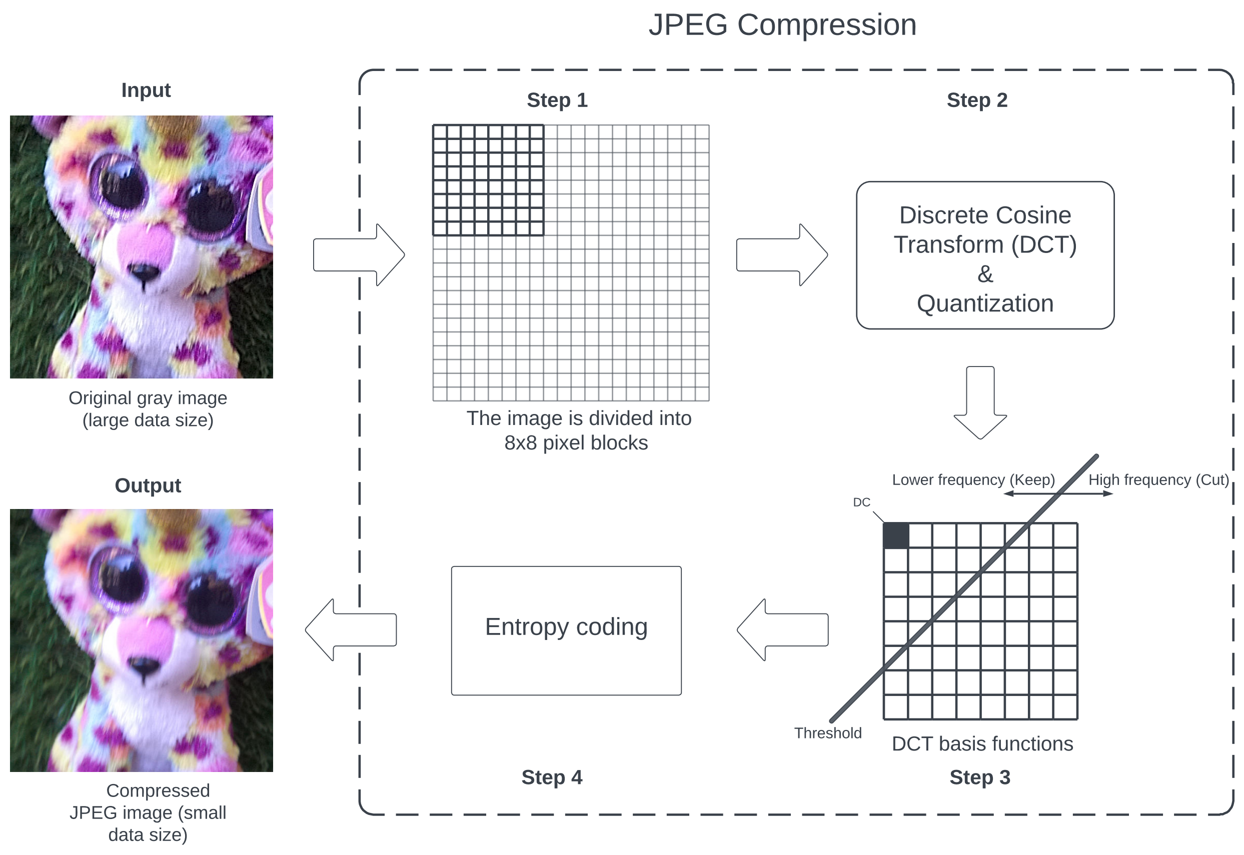


Figure 3.1: The JPEG image compression algorithm [6]. The original image is compressed using a Discrete Cosine Transform calculation and cut the high-frequency component based on a thresholding value (compression ratio).

When a JPEG image is opened, the compressed data is decompressed and the DCT coefficients are inverse-transformed back into the spatial domain, which recreates the original image.

Chapter 4

Steganography algorithms

In the context of steganography, the main goal is to hide information (such as text or images) within other data (usually images or audio files) so that it's not perceptible to a casual observer. There are various techniques that could be used to achieve that. The two primary categories are Spatial Domain techniques and Transform Domain techniques.

The main differences between spatial domain and transform domain steganography techniques are the way they hide information and their robustness against attacks. Spatial domain techniques are simpler and directly modify pixel values, while transform domain techniques involve transforming the image into another domain (such as frequency or wavelet domain) and hiding the data within the transformed coefficients. Transform domain techniques generally provide better security and robustness but can be more complex to implement.

4.1 Spatial domain techniques

These methods involve modifying the least significant bits (LSBs) of the pixel values in the cover image directly. The most common spatial domain technique is the Least Significant Bit (LSB) substitution method. The idea of this approach is to replace the least significant bits of the cover image's pixel values with the bits of the secret message. This method is simple and easy to implement, but it can be more vulnerable to detection and attacks because the hidden data is stored directly in the image.

4.1.1 Least Significant Bit (LSB)

The LSB (Least Significant Bit) method is a steganographic technique for concealing information within digital images by modifying the pixel values' least significant bits. Data is incorporated into the least significant bits of the cover image in this method, making the changes imperceptible to the human eye. As a result, the hidden information in the modified image is visually indistinguishable from the original image.

This technique is simple to understand and it is also quite effective since modifying the last bit of the pixel value does not make any significant changes to the picture. Thus, the original and cover images are not visually different. The Least Significant Bit method for embedding secret text inside involves the following steps:

- **Convert the secret text to binary:** Convert each character of the secret message into its corresponding binary representation, typically using the ASCII or Unicode encoding.

- **Choose a cover image:** Select an appropriate cover image to hide the secret text in. Ideally, the number of pixels of an image should be larger than the length of the binary representation of the hidden data to ensure the hidden data does not significantly impact the image quality.

- **Process the image:** Depending on the image format, the cover image may need to be processed. In our case, the image format is JPEG, which means that it should be converted into an uncompressed format like PNG.

- **Modify the least significant bits:** For each bit of the binary secret message, select a pixel in the cover image and modify the least significant bit of one of the color channels to match the bit of the message. Continue this process for each bit of the message, using a different pixel for each bit.

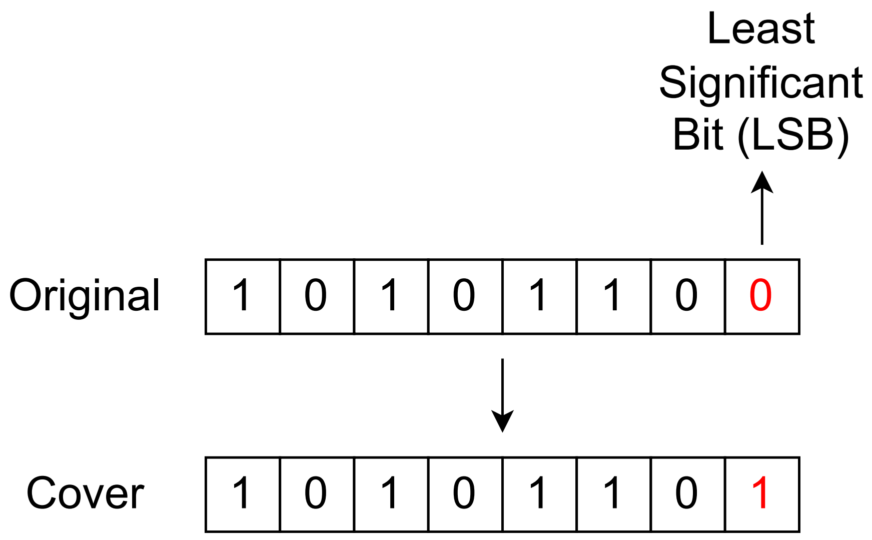


Figure 4.1: The least significant bit of the image is replaced with the bit of the message while preserving the overall structure of the image.

Since this method only replaces the least significant bit of each byte it has a limitation on the maximum payload which could be embedded into the image. However, the main disadvantage of this method is that it is vulnerable to steganalysis and generally not secure. Moreover, the higher the embedded rate, the more vulnerable this method becomes. So the key to preventing successful steganalysis is to keep a low embedding rate.

One of the ways to improve the security is by using modifications to the least significant bit algorithm, such as embedding data only into some pixels in the image by using a particular distribution method that would decide which pixels to use and which to leave out.

4.2 Transform domain techniques

These methods involve transforming the cover image into a different domain (such as frequency or wavelet) and then hiding the secret data within the transformed coefficients. Common transform domain techniques include Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), and Discrete Fourier Transform (DFT). These techniques typically provide better robustness and security compared to spatial domain methods, as they can better resist various image processing operations and attacks. However, they can be more complex and computationally demanding.

4.2.1 J-Uniward

J-Uniward is a content-adaptive method of computer steganography designed for JPEG images. The algorithm was proposed by Vojtěch Holub, Jessica Fridrich, and Tomáš Denemark in their research paper titled "Universal Distortion Function for Steganography in an Arbitrary Domain." [3]. It is an extension of the Universal Wavelet Relative Distortion (UNIWARD) approach, which is a general framework for designing steganographic schemes that minimize embedding distortion in any domain.

J-Uniward builds on the idea of minimizing distortion, using a unique cost function to measure the impact of altering the coefficients of the JPEG image during the embedding process. This cost function, called the J-function, is designed to balance the trade-off between the imperceptibility and robustness of the embedded message. J-Uniward modifies the JPEG coefficients based on this cost function, resulting in a steganographic algorithm that offers improved performance in terms of both visual quality and security.

4.2.2 UERD

The UERD (Universal Embedding for Randomized Distortion) steganography algorithm is a general framework for steganographic schemes that are designed to minimize embedding distortion. The term "universal" in the name refers to the algorithm's adaptability to various distortion measures in different domains. The main goal of UERD is to minimize the detectability of the hidden message by introducing the least amount of distortion while embedding the data.

UERD was proposed by Linjie Guo, Jiangqun Ni, Wenkang Su, Sun Yat-Sen University, Chengpei Tang in their research paper titled "Using Statistical Image Model for JPEG Steganography: Uniform Embedding Revisited." [7]. The algorithm calculates the cost of embedding a message into a cover object, like an image or audio file, and then determines the best way to modify the cover object to minimize distortion. UERD utilizes directional filters to estimate the local smoothness of the cover object, which helps to assess the distortion caused by the embedding process.

- **Input:** A cover JPEG image, a secret message, and a key for embedding.
- **Preprocessing:** Divide the cover image into non-overlapping 8x8 pixel blocks and perform Discrete Cosine Transform (DCT) on each block.
- **Quantization:** Apply quantization to the DCT coefficients, which reduces the number of bits needed to represent each coefficient.
- **Embedding:** Using the key, select certain DCT coefficients and embed the secret message bits into them using a dithering mechanism. The dithering mechanism introduces a small amount of randomness, which helps to make the embedding process less predictable and harder to detect.
- **Dequantization:** Reverse the quantization process on the modified DCT coefficients.

- **Inverse DCT:** Apply Inverse Discrete Cosine Transform (IDCT) to each block of modified coefficients to obtain the stego image.

The UERD algorithm is designed to be secure and robust against various types of steganalysis attacks, as the embedding process is less predictable than traditional uniform embedding methods. This method can determine which regions could be considered noisy and use them to decrease the impact of the embedded data on the statistical features of the image. That reduces the statistical artifacts that can be exploited by steganalysis tools, thus making it more challenging to detect hidden messages in JPEG images.

4.2.3 J-MiPOD

J-MiPOD is a steganography method that can be used to hide information within JPEG-compressed images while minimizing the detectability of the hidden data by a potential eavesdropper. It was presented in the research ”Steganography by Minimizing Statistical Detectability: The Cases of JPEG and Color Images” by Cogranne, Rémi and Giboulot, Quentin and Bas, Patrick [4].

The method is an extension of the MiPOD (Minimizing Probability of Detection) scheme, which is based on the assumption that pixels in an image are statistically independent and follow a Gaussian distribution. In MiPOD, embedding probabilities are determined for each pixel in the image by minimizing the power of the most powerful likelihood ratio test (LRT) for detecting the presence of hidden data.

For JPEG images, J-MiPOD takes into account the fact that the image is compressed using the Discrete Cosine Transform (DCT) and quantization. J-MiPOD first decompresses the DCT coefficients to obtain the original pixel values. It then estimates the variance of the pixel values using a statistical model that takes into account the quantization noise introduced during JPEG compression.

Once the pixel variances have been estimated, J-MiPOD determines the probability of embedding hidden data in each pixel by minimizing the deflection coefficient, which is a measure of how much the embedding changes the statistics of the cover image. The embedding probabilities are then converted into costs, which can be used to determine the optimal locations to embed the hidden data.

Chapter 5

Dataset

A diverse and comprehensive dataset is essential for training an effective steganalysis model, as it allows the model to learn the hidden patterns and artifacts introduced by various steganographic techniques. A well-constructed dataset should include images with hidden data embedded using different steganographic methods, as well as clean images without any hidden information. Furthermore, such a dataset should be large enough, since training deep learning-based steganalysis models require a significant amount of data to learn complex patterns and avoid overfitting.

The ALASKA#2 steganographic dataset [8] is a collection of images specifically designed for the evaluation of steganalysis methods. It is a large-scale dataset that contains both clean images without hidden data and stego images created using various steganographic techniques. The ALASKA#2 dataset was introduced as a part of the Image Steganalysis Challenge in 2020. The dataset provides a large dataset of 80,000 JPEG images from over 40 cameras, including smartphones, tablets, and DSLRs, processed in a realistic and heterogeneous manner.

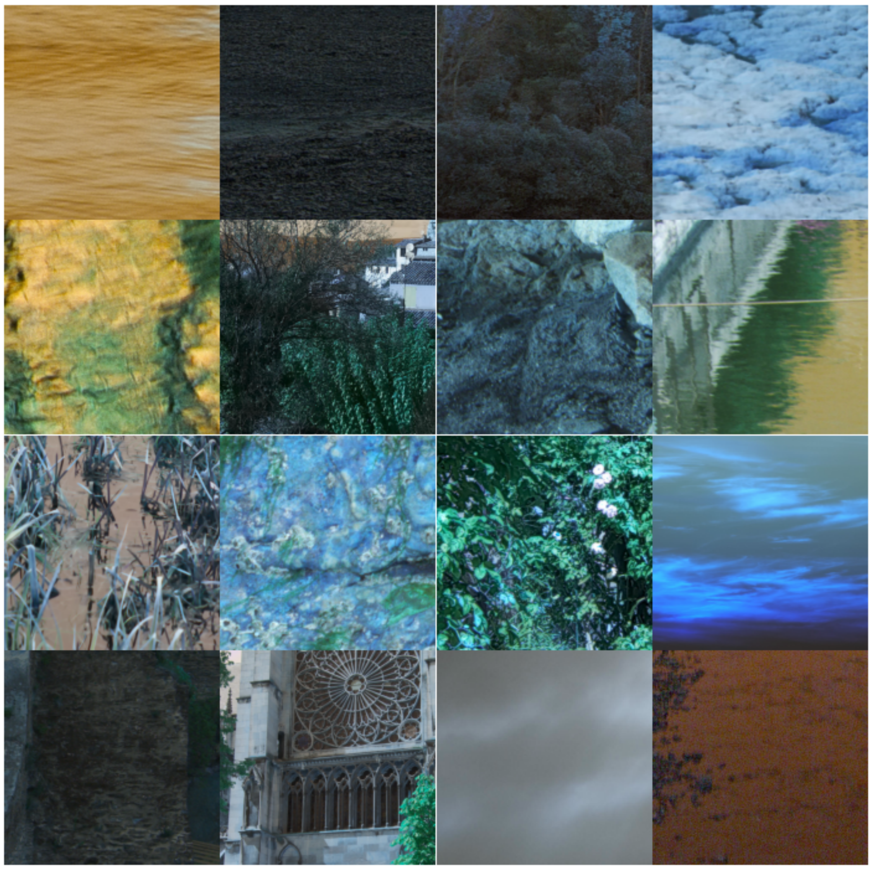


Figure 5.1: Samples of clean images from the dataset.

It used three embedding algorithms: J-UNIWARD, UERD, and J-MiPOD, with an average payload of 0.4 bpnzAC, varying for each image based on complexity. Figure 5.2 demonstrated an example of clean images and a corresponding image with encoded data.

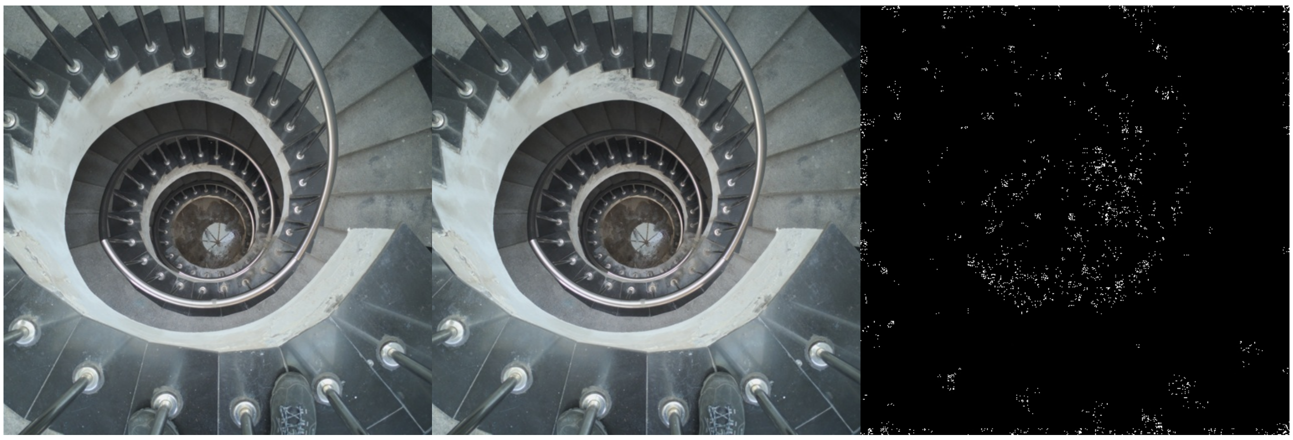


Figure 5.2: Clean image, image with UERD embedded data, scaled differential image between them. The average embedding rate is 0.4 bpnzAC.

The dataset was expanded by incorporating cover images with embedded data using the Least Significant Bit (LSB) algorithm. The addition of LSB-embedded images further broadens the dataset's diversity and enhances the development of steganalysis models, enabling them to better identify and adapt to various data-hiding methods in real-world scenarios.

As a result, the final dataset consists of 75,000 clean RGB images 512x512 and the same number of images for J-UNIWARD, UERD, J-MiPOD, and LSB steganography algorithms, which gives a total of 375,000 images.

Chapter 6

Model

6.1 Data preprocessing

Data preprocessing is an essential step of any deep learning model, since it enhances the quality and consistency of the data, allowing models to learn more effectively, converge faster, and achieve better generalization performance on unseen data.

Firstly, the dataset was normalized by scaling all images between [0, 1]. To improve the model's generalization and prevent overfitting, we also used data augmentation as a part of the preprocessing pipeline. Data augmentation is a technique used in deep learning to increase the amount and diversity of training data without collecting new data. It is also reducing overfitting, and increasing its ability to recognize patterns in unseen data. This was done by applying various transformations to the existing data and creating new, altered versions of the original samples. In particular, we applied the horizontal and vertical flipping techniques with 0.5 probability.

6.2 Architecture

The neural network architecture was based on the EfficientNet architecture, which is a family of convolutional neural networks designed for efficient image classification tasks. It was introduced by researchers at Google Brain in 2019 in their paper ”EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks” [9]. The main goal of EfficientNet is to achieve state-of-the-art accuracy while significantly reducing the number of parameters and computational cost.

The EfficientNet architecture is based on a technique called compound scaling, which involves scaling the network depth, width, and resolution simultaneously. It starts from a baseline model called EfficientNet-B0 and then scales it up to produce a range of larger models, from EfficientNet-B1 to EfficientNet-B7.

In our experiments, the optimal model was the EfficientNet-B2, which has a slightly larger architecture compared to EfficientNet-B1, but still maintains a balance between accuracy and computational efficiency. The weights of the EfficientNet-B2 model are pretrained on a large-scale dataset called ImageNet, which a dataset containing over 1.2 million high-resolution images that span 1000 different object categories. This allows the model to learn rich feature representations and generalizable patterns from a diverse range of images, which significantly speeds up the training process.

6.3 Optimization

We thoroughly developed a method for training our deep learning model by incorporating various advanced methods to improve its performance. To begin with, we used the Adam (Adaptive Moment Estimation) optimizer for training, which is a popular optimization algorithm for training deep learning models.

We used a label smoothing technique for regularization to address the model’s overfitting and enhance its generalization. This method prevents the model from becoming overly confident in its predictions by assigning a small probability to incorrect labels. As a result, the model is encouraged to be more cautious in its predictions, ultimately leading to better performance on unseen data.

Additionally, we implemented the ReduceLROnPlateau scheduler in our training pipeline. This learning rate scheduler monitors a validation loss and dynamically adjusts the learning rate based on the metric's performance. By reducing the learning rate we were able to further optimize our training process, allowing the model to converge to a better solution more efficiently.

6.4 Metrics

In our research, we employed two widely recognized evaluation metrics to assess the performance of our deep learning model, namely the Receiver Operating Characteristic (ROC) Area Under the Curve (AUC) and accuracy. These metrics provide a comprehensive understanding of our model's ability to make accurate predictions and discriminate between different classes.

The ROC AUC metric is a popular evaluation measure, particularly for binary classification problems. It is derived from the Receiver Operating Characteristic (ROC) curve, which is a graphical representation of the true positive rate (sensitivity) against the false positive rate (specificity) at various decision threshold levels. The ROC AUC quantifies the overall performance of a classifier by calculating the area under the ROC curve. A ROC AUC value of 1 represents a perfect classifier, while a value of 0.5 indicates a random classifier.

On the other hand, accuracy is a straightforward and commonly used metric in classification tasks. It is calculated as the ratio of correctly predicted instances to the total number of instances.

6.5 Training

Due to the large size of the original dataset, which would require significant computational resources for training, we decided to reduce it and experiment with several smaller subsets. This allowed us to investigate the impact of dataset size on model accuracy and to identify the minimum dataset size capable of achieving reliable predictions. We organized the deep learning model's training procedure using subsets of different sizes taken from the original dataset, split into the train, test, and validation sets as 70/15/15 accordingly.

Our models were trained on the NVIDIA T4 GPU instances, which were provisioned through the Google Cloud Compute Engine Service. Utilizing high-performance computing resources allowed us to efficiently train large models for a reasonable time.

Figure 6.1 illustrates the relationship between the training time of our models and the size of the dataset, which allows us to better understand the impact of dataset size on the model's training efficiency.

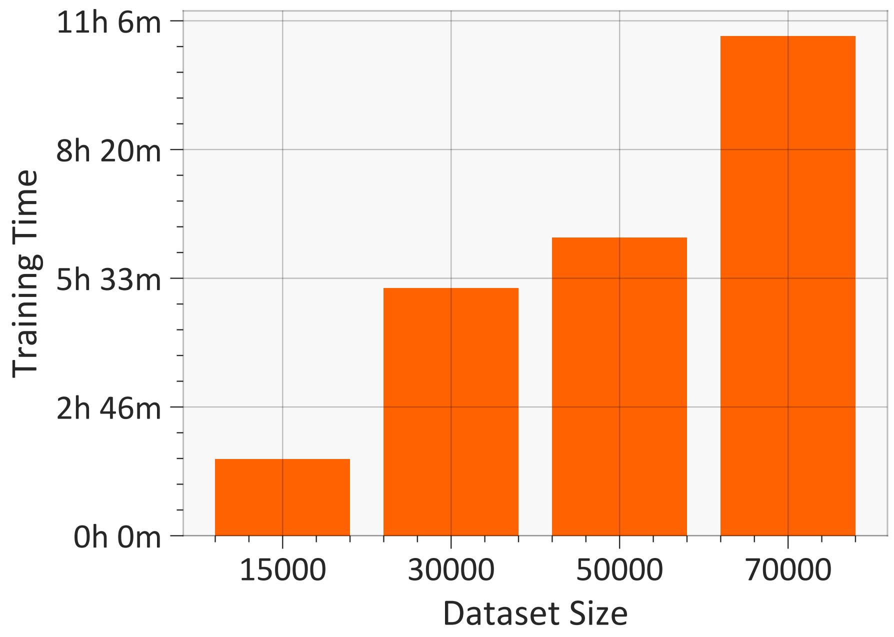


Figure 6.1: The training time is linearly dependent on the dataset size. The training on the subset of 75,000 images took around 11 hours to complete.

The selection of hyperparameters was performed using a systematic process of experimentation and validation. The batch size parameter was chosen taking into consideration the constraints imposed by the GPU memory limit.

The final optimal training hyperparameters are listed in Table 6.1.

Batch size	8
Epochs	15
Learning rate	0.001
Workers	4

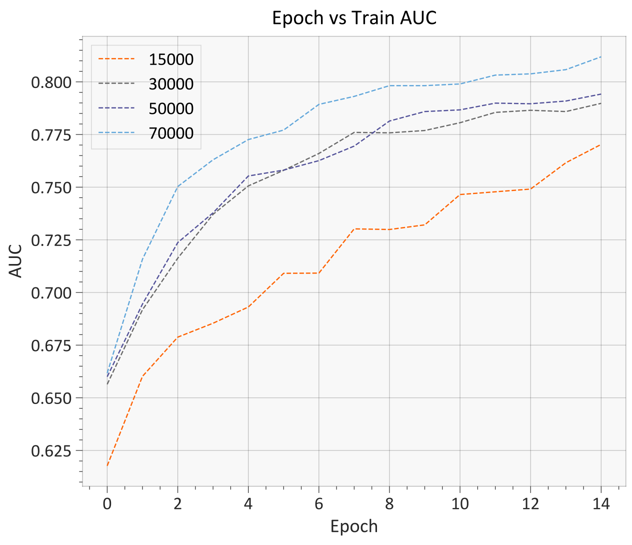
Table 6.1: Training hyperparameters.

Chapter 7

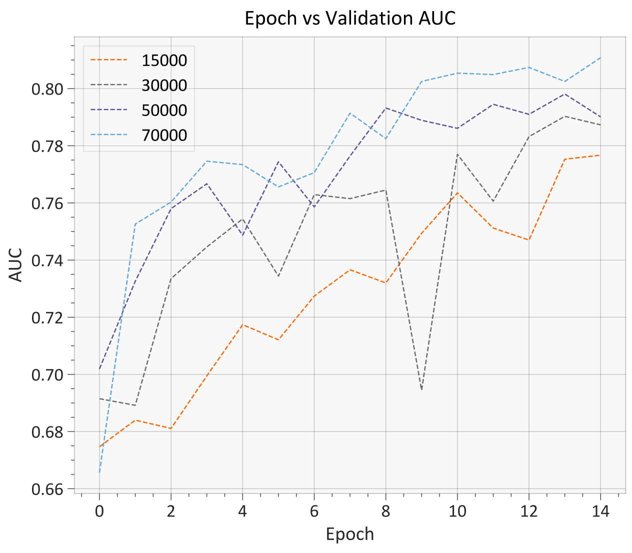
Results

This chapter presents an analysis of the trained models, previously discussed in [10]. In figure 7.1, there are shown the AUC and Loss metrics for four models of different sizes in both training and validation datasets. It is clear that as dataset size grows, the AUC metric improves and the overall dataset loss decreases. The AUC in the training set shows a smooth upward trend, whereas the AUC in the validation set shows a less consistent increase, although the general tendency towards improvement persists.

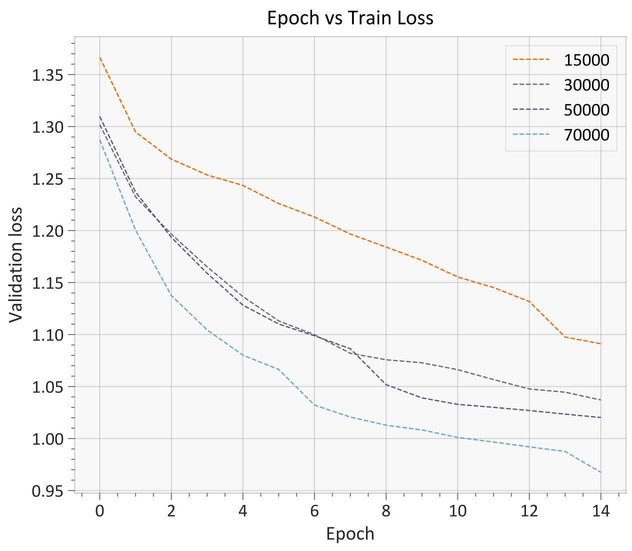
It is important to note that, while larger dataset sizes produce better results in general, the relationship is not strictly linear. The impact on accuracy improvement becomes less pronounced as the amount of data increases, implying that the performance gains gradually diminish with each additional increment in dataset size. Meanwhile, adding more data linearly increases training time, emphasizing the need to strike a balance between the quantity of data and model accuracy. According to our experiments, a minimum of 15,000 images are required to achieve decent accuracy levels.



(a) Train dataset AUC.



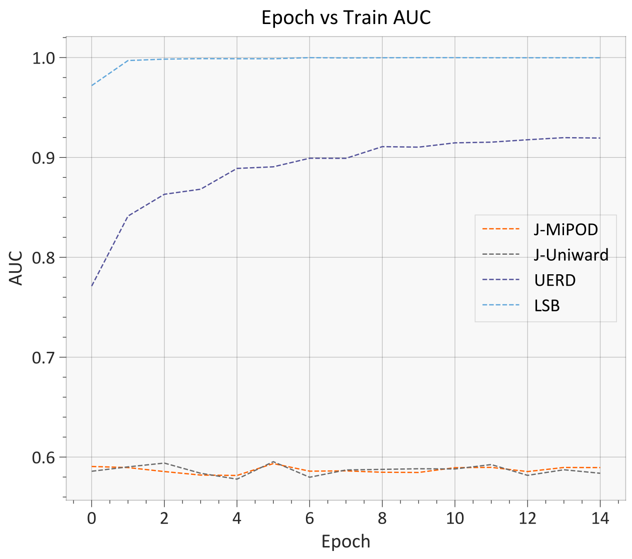
(b) Validation dataset AUC.



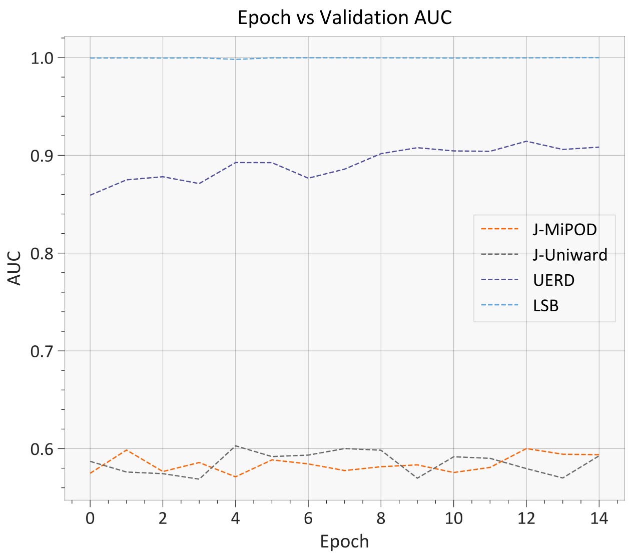
(c) Training loss.

Figure 7.1: General model training metrics.

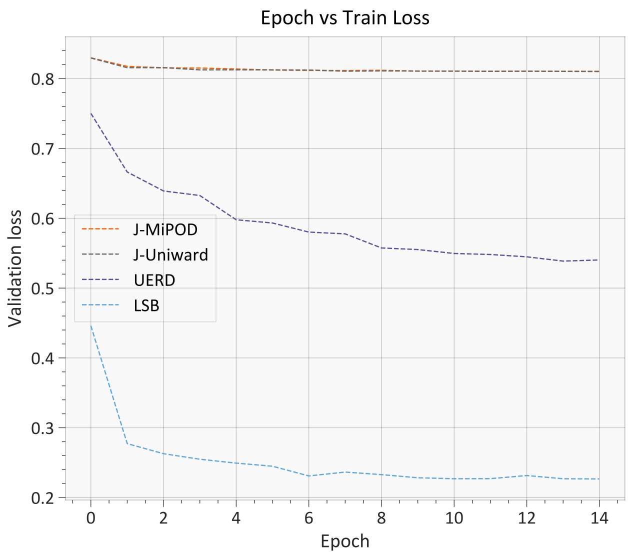
To better understand the effectiveness of each used steganographic algorithm, we trained dedicated models to classify between steganographic images of a particular method and clean images. Figure 7.2 shows the training loss and AUC metrics for these models designed to detect images generated by J-Uniward, UERD, J-MiPOD, and LSB algorithms. Among these models, the LSB detection model demonstrates the highest performance with a ROC AUC score of 0.99. The UERD detection model ranks second, achieving a ROC AUC score of 0.92. However, the detection of steganographic images created by the other two algorithms, J-MiPOD and J-Uniward, proved to be not effective, reaching a maximum ROC AUC score of only 0.6. This indicates high robustness of these algorithms against deep learning steganalysis attacks at the low embedding rate.



(a) Train dataset AUC.



(b) Validation dataset AUC.



(c) Train dataset loss.

Figure 7.2: Training metrics for dedicated models.

Conclusion

In conclusion, this study contributes to the field of steganalysis through the development of a deep learning-based classification model capable of distinguishing between images with and without embedded messages. Our results demonstrate the practical value of the proposed approach, as evidenced by the model's performance in detecting steganographic images generated by various algorithms.

We have shown that larger dataset sizes generally result in better performance. However, the relationship between dataset size and model accuracy is not linear, and the gains in accuracy diminish as the dataset size increases. We have also determined the minimum dataset size required for achieving acceptable accuracy levels, offering practical recommendations for future research in the field.

Our study has further provided valuable insights into the effectiveness of deep learning steganalysis for different steganographic algorithms, revealing the superior performance of LSB and UERD detection models. On the other hand, the models demonstrated low effectiveness on the J-MiPOD and J-Uniward algorithms, which appeared to be robust and almost undetectable at the low embedding rate of 0.4 bpnzAC.

Future research could be focused on designing effective models for low hidden data embedding rates, possibly by exploring advanced feature extraction techniques and investigating alternative deep learning architectures.

References

[1] Jessica Fridrich, Tomáš Pevný, and Jan Kodovský. "Statistically Undetectable Jpeg Steganography: Dead Ends Challenges, and Opportunities". In: *Proceedings of the 9th Workshop on Multimedia ; Security*. MM;Sec '07. Dallas, Texas, USA: Association for Computing Machinery, 2007, pp. 3–14. ISBN: 9781595938572. DOI: 10.1145/1288869.1288872. URL: https://doi.org/10.1145/1288869.1288872.

[2] Mikołaj Płachta et al. "Detection of Image Steganography Using Deep Learning and Ensemble Classifiers". In: *Electronics* 11.10 (2022). ISSN: 2079-9292. DOI: 10.3390/electronics11101565. URL: https://www.mdpi.com/2079-9292/11/10/1565.

[3] Vojtěch Holub, Jessica Fridrich, and Tomáš Denemark. "Universal distortion function for steganography in an arbitrary domain". In: *EURASIP Journal on Information Security* 2014.1 (2014), p. 1. ISSN: 1687-417X. DOI: 10.1186/1687-417X-2014-1. URL: https://doi.org/10.1186/1687-417X-2014-1.

[4] Rémi Cogranne, Quentin Giboulot, and Patrick Bas. "Steganography by Minimizing Statistical Detectability: The Cases of JPEG and Color Images". In: *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security*. IH;MMSec '20. Denver, CO, USA: Association for Computing Machinery, 2020, pp. 161–167. ISBN: 9781450370509. DOI: 10.1145/3369412.3395075. URL: https://doi.org/10.1145/3369412.3395075.

[5] “W3Techs—Web Technology Surveys”. In: (7/May/2023). URL: `https://w3techs.com/technologies/overview/image_format`.

[6] S. Ri et al. “Dynamic Deformation Measurement by the Sampling Moiré Method from Video Recording and its Application to Bridge Engineering”. In: *Experimental Techniques* 44 (Jan. 2020). DOI: `10.1007/s40799-019-00358-4`.

[7] Linjie Guo et al. “Using Statistical Image Model for JPEG Steganography: Uniform Embedding Revisited”. In: *IEEE Transactions on Information Forensics and Security* 10.12 (2015), pp. 2669–2680. DOI: `10.1109/TIFS.2015.2473815`.

[8] Rémi Cogranne, Quentin Giboulot, and Patrick Bas. “The ALASKA Steganalysis Challenge: A First Step Towards Steganalysis”. In: *Proceedings of the ACM Workshop on Information Hiding and Multimedia Security*. IH;MMSec’19. Paris, France: Association for Computing Machinery, 2019, pp. 125–137. ISBN: 9781450368216. DOI: `10.1145/3335203.3335726`. URL: `https://doi.org/10.1145/3335203.3335726`.

[9] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: `1905.11946 [cs.LG]`.

[10] Viktor Seredovych and Yuriy Muzychuk. “Staganalysis using deep neural networks”. In: *International Student Scientific Conference on Applied Mathematics and Computer Sciences (ISSCAMCS)* (May 2023).