TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

Khwopa College Of Engineering
Libali, Bhaktapur
Department of Computer Engineering

A FINAL REPORT ON

IMAGE STEGANALYSIS USING ENSEMBLE CLASSIFIER

Submitted in partial fulfillment of the requirements for the degree

BACHELOR OF COMPUTER ENGINEERING

Submitted by

Sachin Koirala	KCE077BCT029
Sajal Poudel	KCE077BCT031
Unique Shrestha	KCE077BCT045
Utsav Chandra Kayastha	KCE077BCT046

Under the Supervision of
Er. Dinesh Ghemosu
Department Of Computer Engineering

Khwopa College Of Engineering
Libali, Bhaktapur
2023-24

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

Khwopa College Of Engineering
Libali, Bhaktapur
Department of Computer Engineering

A FINAL REPORT ON

IMAGE STEGANALYSIS USING ENSEMBLE CLASSIFIER

Submitted in partial fulfillment of the requirements for the degree

BACHELOR OF COMPUTER ENGINEERING

Submitted by

Sachin Koirala	KCE077BCT029
Sajal Poudel	KCE077BCT031
Unique Shrestha	KCE077BCT045
Utsav Chandra Kayastha	KCE077BCT046

Under the Supervision of
Er. Dinesh Ghemosu
Department Of Computer Engineering

Khwopa College Of Engineering
Libali, Bhaktapur
2023-24

Certificate of Approval

This is to certify that this minor project work entitled **"Image Steganalysis Using Ensemble Classifier"** submitted by Sachin Koirala (KCE077BCT029), Sajal Poudel (KCE077BCT031), Unique Shrestha (KCE077BCT045) and Utsav Chandra Kayashtha (KCE077BCT046) has been examined and accepted as the partial fulfillment of the requirements for the degree of Bachelor in Computer Engineering.

.....
Er. Himal Chand Thapa
Senior Lecturer
External Examiner
Himalaya College of Engineering

.....
Er. Dinesh Ghemosu
Project Supervisor
Senior Lecturer
Department of Computer Engineering
Khwopa College of Engineering

.....
Er. Dinesh Gothe
Head of Department,
Department of Computer Engineering
Khwopa College of Engineering

Copyright

Acknowledgement

We would like to express our sincere appreciation to our supervisor Er. Dinesh Ghemosu, for his invaluable guidance, insightful advice, unwavering help and support throughout the course of this project. His mentorship, suggestions and assistance were instrumental for us. Furthermore, we are deeply grateful for the invaluable contributions of our teachers, Er. Niranjan Bekoju, Er. Mukesh Kumar Pokhrel and Er. Subhadra Joshi. Their expertise, insights and suggestions provided us with different perspectives and solutions to overcome various challenges encountered during the project. Additionally, we would like to express our gratitude to Er. Dinesh Gothe for providing important advice and consistent guidance since the very beginning of the project.
We are also thankful to Dr. Daniel Lerch Hostalot for generously sharing their expertise and providing their guidance and thoughtful feedback. Their collective support and encouragement have been helpful in shaping the outcome of this project, and we are truly grateful for their contributions.

Sachin Koirala	KCE077BCT029
Sajal Poudel	KCE077BCT031
Unique Shrestha	KCE077BCT045
Utsav Chandra Kayastha	KCE077BCT046

Abstract

This project focuses on the detection of steganographically modified images using an ensemble classifier model. Steganography is a technique used to hide information within another carrier file, and it poses a serious threat to digital privacy and security. The proposed model utilizes a bagging method with FLD as base learner to train an ensemble classifier. High dimensional features from the CC-C300 model are used to train the classifier, which offers robustness, model diversity, and faster development compared to other machine learning methods. The model is expected to effectively detect steganographic methods such as nsF5, J-UNIWARD, and UERD, which hide messages in JPEG images. The project is technically feasible, economically feasible, and operationally feasible. The methodology involves data collection from the IStego100K dataset, feature extraction using JPEG coefficient analysis, and training of the ensemble classifier. The expected outcomes include the successful detection of steganographically modified images and the ability to classify them as cover or stego images. The project aims to provide a reliable and efficient solution to the growing threat of steganography in digital communication.

Keywords: Steganography, Steganalysis, Ensemble Classifiers, CC-C300, Machine Learning, FLD , JPEG

Table of Contents

List of Figures

List of Tables

List of Abbreviation

API	Application Programming Interface
bpnzAC	Bits Per Non-zero AC
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
DC-DM	Distortion Compensated-Dither Modulation
DB	Database
DCT	Discrete Cosine Transform
DCTR	Discrete Cosine Transform Residuals
DOM	Document Object Model
FLD	Fisher Linear Discriminant
GBM	Gradient Boosting Machine
GIF	Graphics Interchange Format
GFR	Gabor Filter Residuals
GPU	Graphics Processing Unit
HUGO	Higly Undectectable SteGO
J-Uniward	JPEG UNIversal WAvelet Relative Distortion
JPEG	Joint Photography Experts Group
JSON	JavaScript Object Notation
K-NN	K-Nearest Neighbors
LSB	Least Significant Bit
ML	Machine Learning
OOB	Out-Of-Bag
PHARM	Phase Aware Projection Model
PNG	Portable Network Graphics
RAM	Random Access Memory
STC	Syndrome Trelis Coding
SVM	Support Vector Machine
SQL	Structured Query Language
UERD	Uniform sEmbedding Revisited Distortion
WPC	Wet Paper Codes
YCbCr	Luminance, Chrominance Blue, Chrominance Red

Chapter 1

Introduction

1.1 Background

Steganography is the skillful technique of secret communication, accomplished through embedding a piece of information into a carrier. In steganography, a "carrier" refers to the cover or host file that conceals the hidden message or information. The word steganography is derived from the Greek words *"stegos"* meaning *"cover"* and *"grafia"* meaning *"writing"* defining it as *"covered writing"* [1]. Misusing steganography for malicious purposes can have serious consequences. A recent trend involves exploiting various steganographic techniques to embedd malware in the carrier. Some real life example would be: Hiding malicious code within banner ads, tricking users into installing malicious app, embedding malicious executables and so on. Steganography can utilize various file formats, primarily classified into four categories: text, images, audio, and video. Among these file formats, images are widely used as cover object.
Steganalysis is the process of detecting hidden information within digital media, detecting hidden data that has been secretly embedded using steganographic techniques. The goal of steganalysis is to collect sufficient evidence about the presence of embedded message and to break the security of its carrier. Thus defeating the purpose of steganography. The importance of steganalytic techniques that can reliably detect the presence of hidden information in images is increasing. Steganalysis finds its use in computer forensics, cyber warfare, tracking the criminal activities over the internet and gathering evidence for investigations particularly in case of anti-social elements. [2]

Image Steganography
Image Steganography is a steganography method where the hidden information is embedded into a image as carrier file. Among the various available formats of images, JPEG format is widely used for steganography because of its lossy compression. The figure below shows the glimpse of how a JPEG image is compressed.

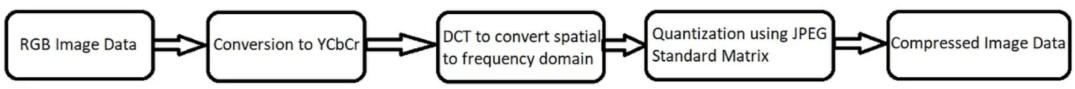


Figure 1.1: Steps used in implementation of Compression Algorithm

First the RGB color space is converted into YCbCr colorspace. After the YCbCr coversion, the image is partitioned into 8×8 non-overlapping blocks. After that each pixel is transfered from range of 0 to 255 to -128 to 127. Finally, Each of these blocks is then subjected to a 2-D Discrete Cosine Transform (DCT). The DCT transforms the spatial data in the block into frequency data. After the DCT, the coefficients are

quantized, meaning they are divided by a factor determined by a quantization table. The quantization step is what actually removes information from the image, and it is this step that makes the JPEG compression process lossy.

$$C(u,v) = \alpha(u)\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)cos[\frac{\pi(2x+1)u}{2N}]cos[\frac{\pi(2y+1)v}{2N}] \tag{1.1}$$

for $u, v = 0, 1, 2, \ldots, N-1$ and $f(x,y)$ is pixel position and $\alpha(u), \alpha(v)$ as

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & for \quad u = 0 \\ \sqrt{\frac{2}{N}} & for \quad u \neq 0 \end{cases} \tag{1.2}$$

$$\alpha(v) = \begin{cases} \sqrt{\frac{1}{N}} & for \quad v = 0 \\ \sqrt{\frac{2}{N}} & for \quad v \neq 0 \end{cases} \tag{1.3}$$

Equation (1.1), (1.2), (1.3) are the formulas to calculate DCT coefficients, $\alpha(u)$ and $\alpha(v)$ respectively. [3]

Some popular techniques of image steganography are listed below:

1. **DCT-LSB Method:**
   The DCT-LSB method conceals data in an image by dividing it into 8x8 blocks, transforming each block using Discrete Cosine Transform (DCT), and then replacing the least significant bit of DCT coefficients with hidden data. [4]

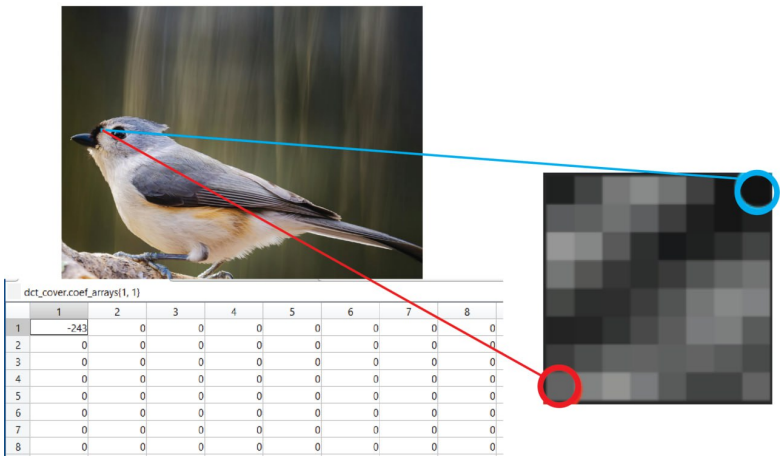
2. **F5 and nsF5 Algorithms:**
   The F5 steganography algorithm operates by manipulating Discrete Cosine Transform (DCT) coefficients. F5 adjusts coefficients by adding 1 to those with a positive value and subtracting 1 from those with a negative value, leaving zero-valued coefficients unchanged. However, a drawback known as the "shrinkage" problem arises when coefficients with values of 1 or -1 become zero during embedding, reducing the algorithm's capacity. [5]
   To overcome the shrinkage problem, the nsF5 algorithm, an advanced version of F5, employs Wet Paper Codes (WPC). This technique designates certain coefficients as "non-modifiable", preventing alterations after data embedding. This innovation addresses the shrinkage issue without sacrificing capacity. Even if new zeros are generated from coefficients with values of 1 or -1, the nsF5 algorithm records them, enhancing image quality and capacity compared to the common F5 method. [6]

3. **UERD (Universal Embedding Reduced Distortion):**
   UERD (Universal Embedding Reduced Distortion) is a method that hides information in pictures so that it's hard to notice.By strategically analyzing Discrete Cosine Transform (DCT) coefficients, UERD chooses those areas in the picture where changes won't be obvious. UERD uses syndrome trellis coding (STC) to hide the message bits in the desired values. This further increases the security of the embedded data by making it harder to detect. [7]

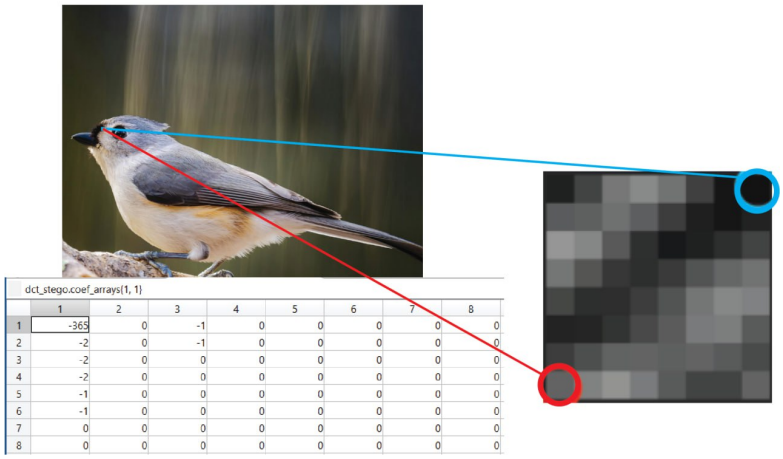
4. **J-UNIWARD Algorithm:**
   The J-UNIWARD algorithm operates in both the spatial domain and the frequency domain. In the spatial domain, it uses the Least Significant Bit (LSB) steganography technique to hide data in the least significant bits of the pixel values. In the frequency domain, it uses the Discrete Cosine Transform (DCT) to transform the image into the frequency domain, where it hides data in the DCT coefficients. [8]

5. **HUGO Algorithm:**
   HUGO, which stands for Highly Undetectable steGO, is a steganography algorithm designed to hide information within images. It achieves this by analyzing complex patterns and structures in an image, determining optimal locations to hide data. [9] [10]

1.2 Comparison of DCT coefficients of cover and stego image



(a) Cover Image



(b) Stego Image

Figure 1.2: cover and stego Image with their DCT coefficients

By comparing the dct coefficient values of the images, we can differentiate between the stego and cover images. For example, the above table displays the respective 8*8 block DCT coefficient table for the cover and stego images.
DCT coefficients represent different frequency components of an image. And in the case of an 8x8 block of pixels given, each block has its set of DCT coefficients. In the example, the first figure represents the statistical features of DCT coefficients from a block of a cover image. If the image is unaltered, the statistical features of its DCT coefficients should exhibit certain expected patterns.
Where as,the second figure represents the DCT coefficients from a block of a stego image. Unusual DCT coefficient value change are compared to the cover image,which doesn't follow the expected pattern indicating that the image is altered and has hidden information.

1.3 Motivation

As motivated by the pressing need to combat cyber threats posed by the continuous evolution of steganography, which presents a serious threat to digital privacy and security, steganalysis tools have been used to detect embedded malicious content in an image. Given the extensive usage of images in digital communication, the use of advanced steganographic techniques in image files expose users to risk by concealing malicious data in easily accessible images. The increasing sophistication of steganographic techniques presents a challenge to traditional steganalysis approaches. This constitutes a significant risk of malicious data infiltration. To overcome this problem, new approaches must be investigated to accurately detect malicious data. By investigating steganalysis using ensemble classifiers, we aim to further strengthen the defenses against sophisticated steganographic attacks while maintaining computational efficiency.

1.4 Problem Statement

The continuous evolution of steganographic techniques poses a critical challenge to digital privacy. With the increasing sophistication of methods used to embed information secretly, traditional steganalysis approaches struggle to accurately detect and mitigate the risks associated with concealed malicious content. As referenced in [11], the computational complexity associated with training SVM is notably high. Moreover, as the number of extracted features increases, the training process using SVM becomes increasingly time-consuming. On the other hand, Ensemble Classifiers offer a faster alternative to SVM while maintaining comparable levels of accuracy. An increasing number of malware infections exploit hidden transmission as cited in [12], reliable steganalysis tools become more important. To address these problems, advanced and flexible steganalysis approach is required. Hence, this project aims to develop steganalysis model using ensemble classifiers to specifically detect hidden data within images.

1.5 Objectives

The main objectives of this project is to:

- To detect steganographically modified JPEG images using Ensemble.

1.6 Scope and application

The major applications of this project are:

- Organizations can employ steganalysis to detect malicious activities such as malware distribution or covert communication channels used by cybercriminals.
- Steganalysis ensures the authenticity and integrity of digital media by detecting any changes or alterations to images.

Chapter 2

Literature Review

Some work has been done in image steganalysis. Various steganalysis tools use different approaches like feature extraction, shallow ML, and deep learning methods to detect steganographically altered images. This literature review seeks to portray the history, methodologies, implementation and applications of steganalysis.

Multiple research has been done to achieve excellent results in steganalysis. Krzysztof Szczypiorski et al. [11] used deep learning and ensemble classifiers to detect image steganography using different methods like DCTR and shallow machine learning classifiers. They found that performance depended heavily on the steganographic method used and on the density of the embedded hidden data. Detection of the content hidden with the nsF5 algorithm at the density 0.4 bpnzac was almost perfect while detection of data hidden using J-Uniward at 0.1 bpnzac was hardly possible. It is shown that steganalysis done using shallow ML is better in comparison to deep learning. This point is further proved by the fact that shallow ML consumes less resources and requires less time to be trained in comparison to deep ML and still provides accuracy similar or better than deep ML classifiers.

The paper [3] gives us a comprehensive overview of the Discrete Cosine Transform(DCT) and its application in digital image and video processing. The document discusses the properties of the DCT, including its decorrelation characteristics, energy compaction, and its ability to reduce entropy. It highlights the DCT's role in efficient coding and compression, particularly in the context of image and video standards such as JPEG and MPEG. Additionally, The document addresses the inverse DCT operation and its impact on visual distortion, providing examples of reconstructed images at different quantization levels.

George Berg et al. [13] proposed an ML approach to steganalysis. This paper shows the feasibility of using a machine learning and data mining (ML/DM) approach to automatically build a steganography attack. This paper used three common data mining and learning techniques: decision trees, error back-propagation, artificial neural networks and the naïve Bayes classifier, to identify messages hidden in compression-(JPEG) and content based (GIF) images.

MT Hogan et al. [14] evaluated the statistical limits by using probability density functions (pdfs). ML tests based on DC-DM are presented in this paper.To effectively uncover hidden information in images, we need a steganalysis tool with sharp pattern recognition skills. Sometimes, when we compare images that have been manipulated with certain tools to their original versions, we can spot a few noticeable visual irregularities – like odd pixels or changes in dimensions due to cropping or padding. If an image doesn't fit specific size criteria, it might get cropped or padded, and you'll see black spaces. Interestingly, most manipulated images don't give away obvious clues when compared to their originals. The simplest clue is a size increase between the manipulated and original images. Other signatures show up in how the colors are arranged in the image, such as a significant change in the number of colors or a gradual increase or decrease. Grayscale images follow a different pattern, increasing incrementally. Another strong indicator is an unusual number of black shades in a

grayscale image.
The paper [15] provides core concepts of this project such as ensemble classifier and importance of selection of features. A distinctive subject which it has touched upon is the concept of Curse of Dimensionality (CoD) which shows the relation of complexity and increase in resource usage for computation. It is highlighted how ensemble classifiers can counter this problem by using reduced dimension for training its base learners.
The paper [16] highlights several key studies in the field of steganalysis, which provides a solid foundation for understanding the current state of steganalysis. The document discusses the implementation of ensemble based steganographically altered image classifier using many base learners for classification. The proposed base learners are trained using FLD analysis due to its ability to increase diversity The performance of the proposed model even though gets trained in very less time in comparison to usually used classification method of G-SVM can classify with similar or better accuracy. It is highlighted that a G-SVM classifier takes about 8 hours to be properly trained while an ensemble classifier takes only 20 minutes.
The paper [17] provides an in-depth insight into the use of an ensemble of classifiers for steganalysis, with a focus on machine learning. The ensemble-based steg analyzer uses feature vectors from multiple stegalyzers to create a decision algorithm that allows the combination of information from different steganalyzers. The resulting steganalyzer is also inherently suitable for multi-class classification scenarios. The paper presents a novel steganalysis decision framework using hierarchical classifiers, which addresses the limitations of existing steganalysis methods and provides a scalable and cost-effective approach to steganalysis. Ensemble classifiers are designed to overcome the limitations of individual classifiers by combining their outputs to achieve better performance. Steganalysis using ensemble classifiers is a powerful approach that utilizes the strength of multiple classifiers to help improve the detection of hidden information in images. It provides diverse steganographic techniques while also enhancing the overall accuracy. Ensemble classifiers are designed to overcome the limitations of individual classifiers by combining their outputs, thereby achieving better performance.
The paper [18] provide information on the pre features and their Cartesian calibrated and Non-cartesian calibrated form.The paper [19] and the paper [20] guides the outlook of our project to a better angle as it provides very crucial details on the section of feature extraction.They provide more insight on the pre features which can be utilized for better classification. These literature provided more insights on CC-PEv and CC-SHI which are different pre features used for steganalysis. “JPEG Image Steganalysis Utilizing both Intrablock and Interblock Correlations” provides more insight on the importance of considering relation between inter and intra block correlations during pre feature creation for better detection or classification.
This Paper [21] explains, IStego 100K is a large-scale steganalysis consisting of 208,104 images with a size of 1024*1024 pixels. The training set consists of 200,000 images organized into 100,000 cover-setgo image pairs. The testing set comprises the remaining 8,104 images. Each image in the dataset has randomly assigned quality factors in the range of 75-95. Three well-known steganographic algorithms J-uniward, nsF5, and UERD [8] [6] [7] are randomly selected for embedding in the images. The embedding rate for each image is randomly set in the range of 0.1-0.4 bpac.

The relevant papers that we studied to grab knowledge about this project are given in the review matrix below:

S.N.	Name	Year	Authors	Dataset	Findings
1	Searching for hidden messages Automatic detection of steganography [13]	2003	George Berg, Ian Davidson, Ming-Yuan Duan, and Goutam Paul	BOSS	Images are commonly used to transmit hidden messages. Different strategies are used to hide messages in GIF and JPEG formats.
2	International Workshop on Information Hiding [21]	2005	Jessica Fridrich, Miroslav Goljan, and David Soukal	NA	matrix LT process,Block Minimal embedding,applications to steganography and data embedding
3	On steganographic embedding efficiency [7]	2007	Jessica Fridrich, Petr Lisonek, and David Soukal.	NA	Matrix embedding using linear codes (syndrome coding) is a general approach to improving embedding effciency of steganographic schemes.
4	Ml detection of steganography [14]	2005	Mark T Hogan, Neil J Hurley, Guenole CM Silvestre, Felix Balado, and Kevin M Whelan	BOSS	Non-blind steganalysis, Distortion-Compensated Dither Modulation
5	The discrete cosine transform (DCT): theory and application [3]	2003	Syed Ali Khayam.	NA	DCT, DCT applications, properties of DCT

S.N.	Name	Year	Author	Dataset	Findings
6	Steganalysis in high dimensions Fusing classifiers built on random subspaces [15]	2011	Jan Kodovsky and Jessica Fridrich.	BOSS	The paper proposes ensemble classifiers as an alternative to support vector machines. Experiments with steganographic algorithms nsF5 and HUGO demonstrate the usefulness of this approach.
7	Ensemble classifiers for steganalysis of digital media [16]	2012	Jan Kodovsky, Jessica Fridrich, and Vojtech Holub.	BOSS	Ensemble classifiers have improved detection accuracy for steganographic methods in JPEG images. The proposed framework allows for fast construction of steganography detectors.
8	Proceedings of the 11th ACM Workshop on Multimedia and Security [18]	2009	Jan Kodovsky and Jessica Fridrich.	NA	covert communication, Digital watermarking
9	Merging markov and dct features for multi-class jpeg steganalysis [20]	2007	Tomas Pevny and Jessica J. Fridrich	BOSS	The paper constructs a new multi-class JPEG steganalyzer with improved performance. The new feature set provides significantly more reliable results compared to previous work.

S.N.	Name	Year	Authors	Dataset	Findings
10	Detection of image steganography using deep learning and ensemble classifiers [11]	2022	Mikołaj Płachta, Marek Krzemien, Krzysztof Szczypiorski, and Artur Janicki	BOSS	Ensemble classifiers performed well in steganography detection. Deep learning algorithms achieved better results for UERD and nsF5 steganographic algorithms.
11	A markov process based approach to effective attacking jpeg steganography [19]	2007	Yun Q. Shi, Chunhua Chen, and Wen Chen.	BOSS	The proposed steganalyzer outperforms by a significant margin. The detection rates are higher while considering the introduced features.
12	A fast and accurate steganalysis using ensemble classifiers [17]	2013	Arezoo Torkaman and Reza Safabakhsh	BOSS	The proposed method achieved a lower error rate of 46% compared to the ensemble classifier. The training time of the proposed method was 88% lower than the ensemble classifier.
13	Istego100k Large-scale image steganalysis dataset [21]	2019	Zhongliang Yang, Ke Wang, Sai Ma, Yongfeng Huang, Xiangui Kang, and Xianfeng Zhao	IStego100K	The paper introduces a large-scale image steganalysis dataset called IStego100K. The performance of some steganalysis algorithms on IStego100K is tested.

Table 2.1: Review Matrix with Research Papers, authors and purpose

Chapter 3

Requirements Analysis

3.1 Software Requirements

Software requirements for the given project are as follows:

1. **Python:** Python is a versatile programming language commonly used for developing software applications. It can be used for various tasks in the system, such as backend development, data processing, and machine learning integration. ementing interactive features on the system's web interface, and facilitating communication with the backend.
2. **GitHub:** GitHub is a web-based platform for version control using Git. It provides a collaborative environment for software development projects, allowing developers to host and share their code, track changes, manage workflows, and collaborate with others. GitHub offers features such as code repositories, issue tracking, project management tools, code review, and continuous integration.
3. **MATLAB:** MATLAB, short for "Matrix Laboratory," is a high-level programming language and interactive environment primarily used for numerical computation, visualization, and programming. It provides a wide range of built-in functions and toolboxes for various applications, including mathematics, signal processing, image processing, control systems, and machine learning.
4. **VS Code:** VS Code is a popular and widely used source code editor that offers a range of features and extensions to enhance the development experience. It supports multiple programming languages, including Python, JavaScript, and React, making it suitable for working with the different components of the system.
5. **LaTeX:** LaTeX is a typesetting system used for creating documents, particularly those that require complex mathematical equations or scientific notation. It is widely used in academic and technical fields for its ability to produce high-quality documents with consistent formatting. LaTeX uses markup language to format text, and it is highly customizable, allowing users to create templates and styles for their documents. It is also free and open-source, making it accessible to anyone who wants to use it.

3.2 Hardware Requirements

Hardware requirements for the given project are as follows:

1. Employed systems equipped with 16 GB RAM, 3.6 GHz processors, and NVIDIA GPUs for optimal performance.
2. Utilized a dedicated server having 64 GB RAM to enhance computational capabilities.

3.3 Functional Requirements

The functional requirements for the prepared project are as follows:

1. The Final model must be able to distinguish between cover and stego images.
2. The system must be able to process images in real-time.
3. The UI must be user-friendly and responsive.

3.4 Non-Functional Requirements

The non-functional requirements for the prepared project are as follows:

1. **Reliability:** The system must be reliable and consistent in its performance.
2. **Maintainability:** The system must be easy to maintain and update.
3. **performance:** The system must be able to process images quickly and efficiently.
4. **Accuracy:** The system must be able to accurately detect stego images.
5. **Compatibility:** The system must be compatible with different operating systems and devices.

Chapter 4

Feasibility study

4.1 Technical Feasibility

For the technical part, we obtained our project data from the IStego100K [21] dataset, which contained 200,000 images. These images had been modified using three different algorithms, which created diversity in the dataset used, improving the reliability of the system. We used free software to build the project, and for the training and testing, we used our own computers. Thus, we concluded that it was technically feasible.

4.2 Economical Feasibility

The only cost for the project is the computational power, covering processing and electricity. For development of the system, all the softwares used are free of cost and hardwares were available for us to train model.

4.3 Operational Feasibility

We had decided to use the Shallow ML approach, which allowed the model to be trained with less computational power in comparison to deep learning. For shallow machine learning, we planned to implement an ensemble classifier, and each of its models was trained using FLD to improve its effectiveness. Deep learning implemented the CNN approach, which required higher computational power to be trained. Thus, we decided to use a simpler machine learning approach that didn't need a lot of computational power, unlike the more complex deep learning method called Convolutional Neural Network (CNN). This way, the system was practical and didn't need a lot of resources, making it able to be effectively implemented in real-life applications, making it operationally feasible.

4.4 Schedule feasibility

The project was scheduled to be completed within in 6 months duration. Our project completed as scheduled in Gantt chart.

4.5 Gantt Chart

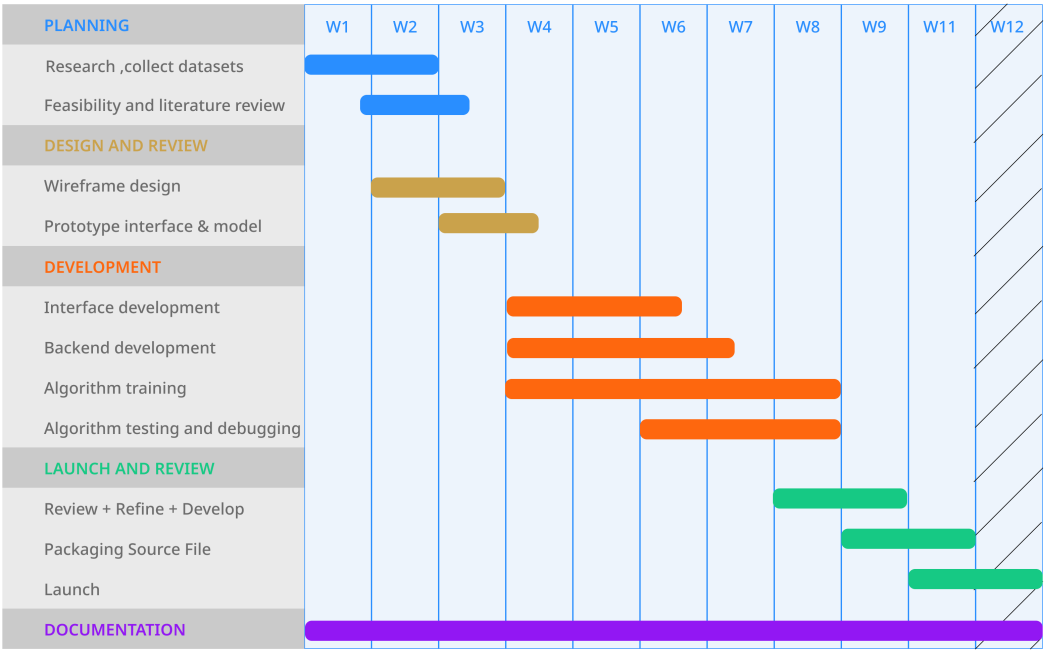


Figure 4.1: Gantt Chart

Chapter 5

Methodology

5.1 Data Collection

We have gathered data from the IStego100K dataset [21], a comprehensive collection of images for steganalysis research. The training images in IStego100K were collected from Unsplash[1], a copyright free photography website. This dataset consits 200,000 JPEG images, divided into two main categories: ”Stego” and ”Cover,” each containing 100,000 images. The ”Stego” subset consists of images where the original cover images from the dataset have been modified to hide information. Each steganographic image is randomly steganized with three widely used image steganography algorithms (J-uniward [8] ,NSF5 [5] and UERD [7]) with a random embedding rate (0.1-0.4bpac). This diversity makes the dataset particularly valuable for developing models capable of detecting hidden data across various embedding techniques and makes dataset universal. Conversely, the ”Cover” subset consists of unaltered images, serving as the baseline for comparison.
The images in the dataset are standardized to 1024x1024 pixels in RGB format, with each image assigned random quality factors ranging from 75 to 95. Additionally, the payload for embedding is randomly distributed within the range of 0.1 to 0.4 bits per coefficient (bpac).[0.10 bpac means that for every cofficient in the DCT domain of the cover image, 0.10 bits of secret data are hidden on average.]
There are alternative datasets for steganalysis, such as BOSSbase 1.01 and BURSTbase. However, these datasets utilize images in .pgm format and we felt difficult to work with images in .pgm formatt. We preferred for the IStego100K dataset due to its utilization of .jpg format. Additionally, while the former datasets employ a single steganographic algorithm for embedding information, IStego100K employs three different algorithms, enhancing its diversity and utility for research purposes. Downloading the IStego100K dataset was much easier because it was available on Google Drive, whereas the other datasets were harder to find and access. Additionally, the information provided about the IStego100K dataset was clearer compared to the limited details available for the other datasets. Another advantage of IStego100K is that it’s newer, which suggests it might be more up-to-date and relevant for current research compared to the older datasets.

[1] https://unsplash.com/

5.2 Dataset Pre-processing

The pre feature space is the space before the feature extraction process which involves various processes such as normalization, noise reduction or other process to enhance the quality of the data before feature extraction. The images were changed or modified as the need arose ,in order to prepare it for feature extraction process. The feature extraction process we were to perform was CC-C300 which took features from only one channel from the RGB thus, to ensure that all the information from image is considered on a single channel we turned the colored image into gray scaled image which eliminates color information, retains only the luminance (brightness) channel while still maintaining the important structural details and intensity variations from the original colored image. When conversion of colored image to grayscale is done, the colors are lost, but the grayscale version still keeps important stuff like the differences in brightness, edges, and the overall structure of the image.

5.3 Feature extraction

5.3.1 JPEG Coefficient Analysis

In the initial step, the JPEG coefficients for each 8×8 block are computed, denoted as $D(i,j)(k,l)$, representing the (k,l) coefficient in the i,j block. Here, $k,l = 0,1,2,...7$, $i = 1,2,3,...,8(M/8)$, and $j = 1,2,3,...,8(N/8)$, with $M \times N$ as image dimensions. Subsequently, the coefficients undergo truncation to simplify complexity, ensuring the quantized DCT or JPEG coefficients are centered around 0. The distribution of the differences between JPEG coefficients exhibits a Laplacian-like pattern, with most values close to zero. Truncation is applied to all available JPEG coefficients, expressed as $D(i,j)(k,l) \leftarrow \text{trunc}_T(D(i,j)(k,l))$, where $\text{trunc}_T(x) = x$ for $x \in [-T,T]$.

$$P(\Delta i, \Delta j, k_1, l_1, k_2, l_2) = \left\{ \left[D^{(i,j)}_{k_1 l_1}, D^{(i+\Delta i, j+\Delta j)}_{k_2 l_2} \right] | i = 1,...,N^{(i)}, j = 1,...,N^{(j)}) \right\}$$

Following truncation, the values are processed to form pairs, and a co-occurrence matrix is computed based on the condition $|\triangle(i)| + |\triangle(j)| + |k_1 - k_2| + |l_1 - l_2| > 0$, ensuring distinct coefficients are utilized. Here, $\triangle(i) = i_1 - i_2$ and $\triangle(j) = j_1 - j_2$. The resulting co-occurrence matrix is derived from the truncation process, generating a transition probability matrix with a dimensionality of $(2T+1) \times (2T+1)$. For instance, if $T = 4$, the co-occurrence matrix takes on a dimension of 81 for every pair of DCT modes denoted by $(\triangle(i), \triangle(j), k_1, l_1, k_2, l_2)$.

$$Cst = \frac{1}{N^{(i)} N^{(j)}} \sum_{i=1}^{N^{(i)}} \sum_{j=1}^{N^{(j)}} [a,b] \in P(\Delta i, \Delta j, k1, l1, k2, l2) | a = s, b = t$$

5.3.2 Truncation

The co-occurrence matrix is calculated between each pair as depicted in img 2. The truncation process yields a transition probability matrix with a dimensionality of $(2T + 1) \times (2T + 1)$. When the threshold value (T) is set to 1, considering elements within the range $\{-1, 0, 1\}$, it results in a transition probability matrix with a dimensionality

of 3×3. Let's denote the states as -1, 0, and 1, and assume a hypothetical transition probability matrix P:

$$P = \begin{bmatrix} p_{-1,-1} & p_{0,-1} & p_{1,-1} \\ p_{-1,0} & p_{0,0} & p_{1,0} \\ p_{-1,1} & p_{0,1} & p_{1,1} \end{bmatrix}$$

In this matrix: - $p_{-1,-1}$ represents the probability of transitioning from state -1 to state -1. - $p_{-1,0}$ represents the probability of transitioning from state -1 to state 0. - Similarly, the other entries follow the same pattern for transitions between states 0 and 1.
The truncation value T is set to 4, resulting in a co-occurrence matrix of dimension 81 for every pair of DCT modes represented by $(\triangle(i), \triangle(j), k_1, l_1, k_2, l_2)$.
Initially, all possible DCT mode pairs are sorted based on their importance, and co-occurrences are then formed from the most important to the least important ones. As a measure of importance, mutual information (MI) is computed over a sufficiently large set of randomly selected DCT coefficient pairs from those two modes, using the `mutual_info_score(mode1, mode2)` where `mode1` and `mode2` are selected randomly.

5.3.3 Feature Set Creation

The top N concatenated co-occurrence matrices, denoted as CN, yield a dimension of $N \times 81$. After Cartesian calibration, the dimensionality of what we denote as $CC - CN$ doubles to $2 \times N \times 81$. Choosing N as 300, we have $CC - C300$ with a dimensionality of $2 \times 300 \times 81 = 48,600$ for each image, irrespective of image dimensionality.
For the concatenated matrix, let

$$C_1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad C_2 = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}, \quad C_3 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

For $CC - C2$, the importance of each co-occurrence matrix is determined by mutual information (MI). If C_1 and C_2 are of highest importance, a new matrix is formed by concatenating C_1 and C_2.

$$C = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix},$$

where the dimension is $2 \times 3 \times 3 = 18$.
For mutual information (MI) to calculate the importance of co-occurrence, entropy and conditional entropy for each feature are calculated. The mutual information is then computed as the difference between entropy and conditional entropy for each feature, given by mutual_info_score(mode1, mode2).

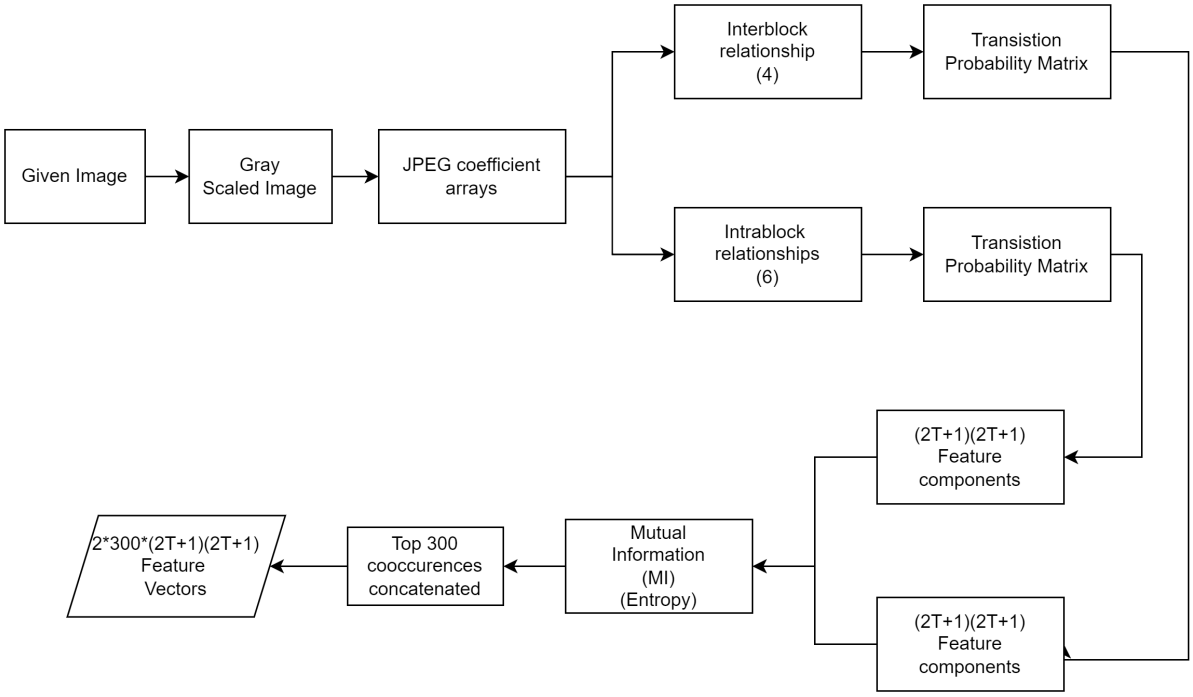
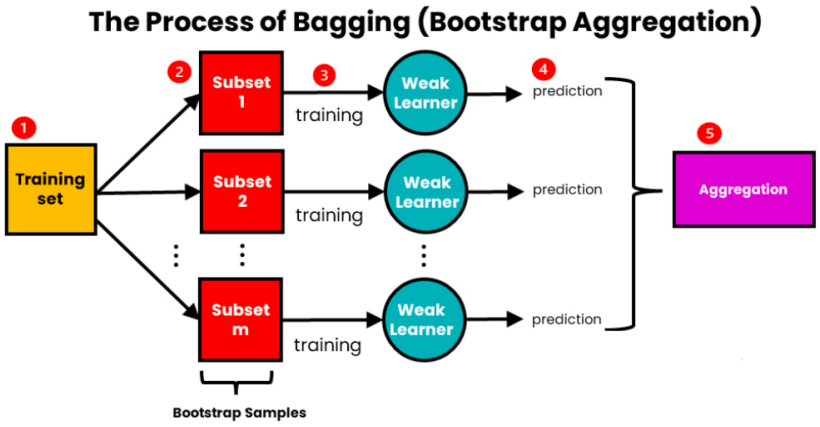


Figure 5.1: Feature Extraction

5.4 Ensemble Classifiers

Ensemble classifiers are machine learning techniques that combine multiple individual models, or "base learners," to make predictions. The main idea behind the ensemble methodology is to weigh several individual classifiers, and combine them in order to obtain a classifier that outperforms every one of them. The key objective of the ensemble methods is to reduce bias and variance. High bias means the model is too simplistic and may not capture the underlying patterns in the data, leading to underfitting. Ensemble methods aim to reduce bias by combining multiple models, each capturing different aspects of the data. High variance means the model is overly complex and may capture noise or random fluctuations in the data, leading to overfitting. Ensemble methods aim to reduce variance by averaging the predictions of multiple models, smoothing out individual fluctuations and creating a more stable and reliable overall prediction.
Some of widely used ensemble methods are:

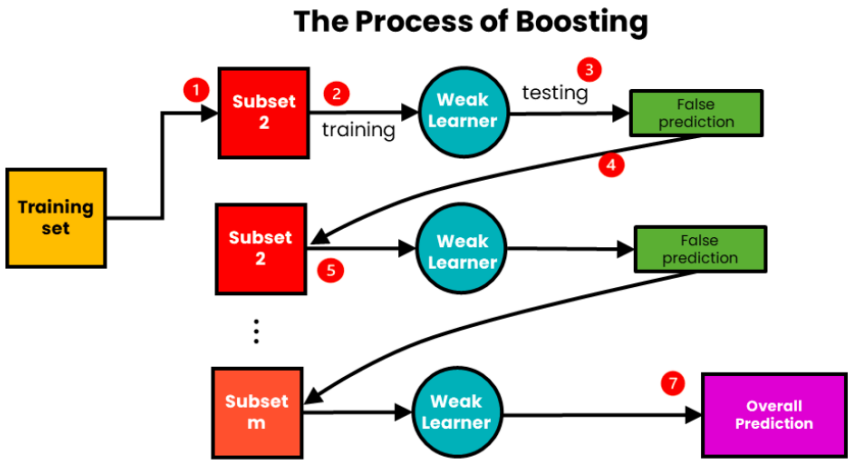
- **Bagging:** Bagging, or Bootstrap Aggregating, is an ensemble method that involves training multiple models on different subsets of the training data and combining their predictions. The subsets are created by sampling the training data with replacement, and each model is trained on a different subset. The final prediction is made by averaging the predictions of all the models. Bagging helps reduce variance by creating diverse models and averaging their predictions, leading to a more stable and reliable overall prediction.



Source: https://www.analyticsvidhya.com/blog/2023/01/ensemble-learning-methods-bagging-boosting-and-stacking/

Figure 5.2: Basic outline of Ensemble Classifier

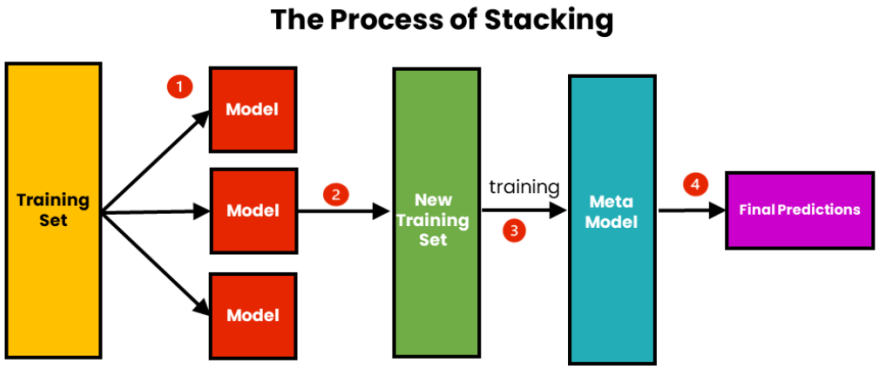
- **Boosting:** Boosting is an ensemble method that involves training multiple models sequentially, with each model learning from the mistakes of the previous models. The final prediction is made by combining the predictions of all the models. Boosting helps reduce bias by focusing on the mistakes made by the previous models and learning from them.



Source: https://www.analyticsvidhya.com/blog/2023/01/ensemble-learning-methods-bagging-boosting-and-stacking/

Figure 5.3: Basic outline of Boosting Classifier

- **Stacking:** Stacking is an ensemble method that involves training multiple models and combining their predictions using another model, called a meta-learner. The meta-learner is trained on the predictions of the base models and learns to combine them into a final prediction.



Source: https://www.analyticsvidhya.com/blog/2023/01/ensemble-learning-methods-bagging-boosting-and-stacking/

Figure 5.4: Basic outline of Stacking Classifier

A standard ensemble approach used in classification tasks consists of the following fundamental components:

1. **Training set:** A labeled dataset is used for ensemble training. The instances of the dataset are described as attribute-value vectors. This set is crucial as it serves as the foundation for training the ensemble model, providing the necessary data points and their corresponding labels. In other words, it's the collection of data points used to train an ensemble learning algorithm. Each data point in the training set consists of two parts: the features (or attributes) and the label (or output). The features are the input variables used to predict the label, and the label is the output variable we want to predict.

2. **Base Inducer:** The inducer is an induction algorithm that obtains a training set and forms a classifier that represents the generalized relationship between the input attributes and the target attribute. Decision trees, Neural networks, Support Vector Machines(SVMs), k-Nearest Neighbors(k-NN), Random Forsets, Gradient Boosting Machines(GBMs), AdaBoost etc. are different types of base learners that can be used in ensemble methods. The choice of base learner plays a vital role in the ensemble's performance and is often selected based on the nature of the problem and the characteristics of the data.

3. **Diversity Generator:**This component is responsible for generating the diverse classifiers. Diversity among the classifiers is crucial as it helps improve the ensemble's performance by ensuring that the individual models provide different perspectives on the data. Techniques such as bagging, boosting, and randomization can be used to generate diverse classifiers.

4. **Combiner:** The combiner is responsible for combining the classifications of the various classifiers. Some of the widely used combination methods are Weighting method, Majority voting, Performance weighting, Distribution summation, Bayesian combination, Dempster-Shafter, Vogging, Naive bayes. The choice of combination method depends on the nature of the problem and the characteristics of the data. The combiner's role is to aggregate the predictions of the individual models to produce the final ensemble prediction, which often leads to improved overall performance compared to using a single model. [22]

5.4.1 Algorithm

Algorithm 1 Ensemble Classifier Algorithm [16]

for $l = 1$ to L **do**
  Form a random subspace
  $D_l \subset \{1, \ldots, d\}, |D_l| = d_{\text{sub}} < d$
  Form a bootstrap sample N_1^b, $|N_1^b| = N^{\text{trn}}$ by uniform sampling with replacement from the set $\{1, \ldots, N^{\text{trn}}\}$
  Train a base learner B_l on features
  $X_l = \{x_m^{(D_l)}, \bar{x}_m^{(D_l)}\}_{m \in N_l^b}$
  $\rightarrow$ obtain eigenvector v_l and threshold T_l
end for
for all $y \in Y^{\text{tst}}$ **do**
  for $l = 1$ to L **do**
    Make l^{th} decision: $B_l(y^{D_l}) \triangleq \begin{cases} 1, & \text{when } v_l^T y^{(D_l)} > T_l \\ 0, & \text{otherwise} \end{cases}$
  end for
end for
Form the final decisions $B(y)$ by majority voting:
$B(y) = \begin{cases} 1, & \text{when } \sum_{l=1}^{L} B_l(y^{(D_l)}) > L/2 \\ 0, & \text{when } \sum_{l=1}^{L} B_l(y^{(D_l)}) < L/2 \end{cases}$
return $B(y)$, $y \in Y^{\text{tst}}$

In the provided algorithm:

d: Represents the dimensionality of the feature space.
d_{sub}: Represents the dimensionality of the feature subset.
N^{TRN} and N^{TST}: Denote the number of training and testing examples, respectively.
L: Represents the number of base learners.
$x_m, \bar{x}_m \in \mathbb{R}^d, m = 1, ..., N^{\text{TRN}}$: Refer to the cover and stego features computed from the training set.
$y_k, \bar{y}_k \in \mathbb{R}^d, k = 1, ..., N^{\text{TST}}$: Denote the cover and stego features computed from the testing set.

5.5 Model Training Approach

Ensemble Classifier Selection:
The decision for selecting an ensemble classification approach is based on the problem at hand. The bagging approach appears ideal for this classification problem as it effectively addresses issues such as overfitting, prevalent during model training. One of the main focuses of our model is diversity, and bagging achieves this by creating diverse bootstrapped samples, enhancing the accuracy of the classification. Bagging also allows parallelization, significantly reducing training time compared to other models like boosted ensemble classifiers, L-SVM, or G-SVM. The simplicity and instability of bagging contribute to its advantages for ensemble classification.

Bootstrapping:
Bootstrapping is simply the process of randomly dividing the dataset into various parts and feeding them to the number of base learners present inside the ensemble classifier for training. The datasets are divided into various smaller samples and then sent to the base learners for training. This division of datasets removes their dependency on each other, promoting parallelization and allowing the training of various models simultaneously..

Base Learner Training:
The base learners are fed with the feature ”dsub,” a subset of ”d” where ”d” is the total number of features or dimensions of each data. The base learners are trained on Fisher's Linear Discriminant Analysis (FLD), which outputs binary classification. FLD is chosen for its diversity in revealing various errors, increasing the diversity of each base learner and resulting in a more accurate ensemble classifier.

Aggregation:
The trained model can now successfully classify the input images as cover or steganographically modified images. The binary classifier typically classifies 1 as a steganographically modified image and 0 as a cover image. The chosen voting method is hard voting, which calculates the number of 0s and 1s and outputs the result dominated by the majority. The threshold is set at L/2.

Efficiency Considerations:
The system prioritizes efficiency by embracing shallow machine learning techniques, specifically ensemble classifiers, instead of deep learning approaches. This strategic choice aligns with the computational efficiency goal, ensuring effective steganalysis without the computational demands associated with deep learning architectures. The intentional selection of CC-C300 as a feature further amplifies efficiency and bolsters the system's detection capabilities. This streamlined approach aims to strike a balance between accuracy and efficiency in steganalysis.

5.6 FLD as Base Learner

Fisher's Linear Discriminant (FLD), also known as Linear Discriminant Analysis (LDA), is a popular method used for dimensionality reduction and classification. It was introduced by Ronald A. Fisher in 1936 and is based on the idea of finding a linear combination of features that characterizes or separates two or more classes of objects or events. FLD is a supervised learning algorithm, meaning it requires labeled data to train the model. It is commonly used for classification tasks, where the goal is to predict the class label of a new data point based on its features. FLD works by finding the linear combination of features that maximizes the separation between the classes while minimizing the variation within each class. This linear combination is known as the discriminant function, and it is used to classify new data points. FLD is particularly effective when the number of features is larger than the number of samples. FLD is widely used in various fields, including pattern recognition, machine learning, and statistics, and it has been successfully applied to a wide range of classification problems. Modern steganographic methods typically require a high dimensional feature representation of images for accurate detection. FLD is well-suited for this task and is a widely used machine learning tool for steganalysis of digital media due to its efficiency when working with high dimensional feature sets. It is also known for its ability to handle multicollinearity, a common issue in high dimensional feature sets. [23]

5.6.1 FLD algorithm for Image steganalysis

1. **Input:**
   - Training sets of cover and stego image features (matrices of size $d \times N_{\text{trn}}$)
   - Number of base learners (L)
   - Dimensionality of the feature space (d)
   - Number of features selected for each base learner (d_{sub})
2. **Calculate the means (μ_c and μ_s) and covariance matrices (Σ_c and Σ_s) of the cover and stego image features.**
3. **For each base learner:**
   a. Randomly select d_{sub}-dimensional subsets of the feature space.
   b. Calculate the weighting vector (w) using the Fisher Linear Discriminant (FLD) criterion:
      - Calculate the means (μ_c and μ_s) and covariance matrices (Σ_c and Σ_s) for the selected features.
      - Calculate the weighting vector (w) using the formula: $w = (\Sigma_c + \Sigma_s)^{-1}(\mu_c - \mu_s)$
4. **Combine the decisions of all base learners:**
   a. For each test feature vector (f):
      - Calculate the projections (v) of the feature vector onto the weighting vectors of all base learners.
      - Combine the decisions of all base learners using a majority voting rule or other fusion method.

5. **Output:** The final classification decision for the test feature vector.

5.6.2 Fundamentals of Linear Discriminant Analysis

- **Class Mean:**
  - Calculate the mean vectors for each class:

$$m_1 = \frac{1}{N_1} \sum_{i=1}^{N_1} x_i^{(1)}$$

$$m_2 = \frac{1}{N_2} \sum_{i=1}^{N_2} x_i^{(2)}$$

    Where N_1 and N_2 are the number of samples in class C_1 and C_2 respectively, and $x_i^{(1)}$ and $x_i^{(2)}$ are feature vectors for individual samples.

- **Within-Class Scatter Matrix:**
  - Calculate the within-class scatter matrix S_w:

$$S_w = \sum_{i=1}^{N_1} (x_i^{(1)} - m_1)(x_i^{(1)} - m_1)^T + \sum_{i=1}^{N_2} (x_i^{(2)} - m_2)(x_i^{(2)} - m_2)^T$$

- **Between-Class Scatter Matrix:**
  - Calculate the between-class scatter matrix S_b:

$$S_b = (m_1 - m_2)(m_1 - m_2)^T$$

- **Fisher's Criterion:**
  - Find the optimal weight vector W by maximizing the Fisher's criterion:

$$W = S_w^{-1}(m_1 - m_2)$$

    Where S_w^{-1} is the inverse of the within-class scatter matrix.

- **Decision Rule:**
  - The decision rule for classifying a new feature vector x is determined by:

$$\text{if } W^T x > \text{Threshold, then classify as } C_1$$
$$\text{if } W^T x \leq \text{Threshold, then classify as } C_2$$

5.7 Model Evaluation

5.7.1 Performance Metrics

5.7.1.1 Precision

Precision, also known as positive predictive value, is the fraction of retrieved instances that are relevant. It is calculated using the formula:

$$Precision = \frac{TP}{TP + FP}$$

5.7.1.2 Accuracy

Accuracy is the ratio of correctly predicted observations to the total observations. The formula for accuracy is:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

5.7.1.3 Recall

Recall, also called sensitivity, is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. It is calculated using the formula:

$$Recall = \frac{TP}{TP + FN}$$

5.7.1.4 F1 Score

F1 Score is the weighted average of Precision and Recall. The formula for F1 Score is:

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Chapter 6

System Design and Architecure

6.1 Use Case Diagram

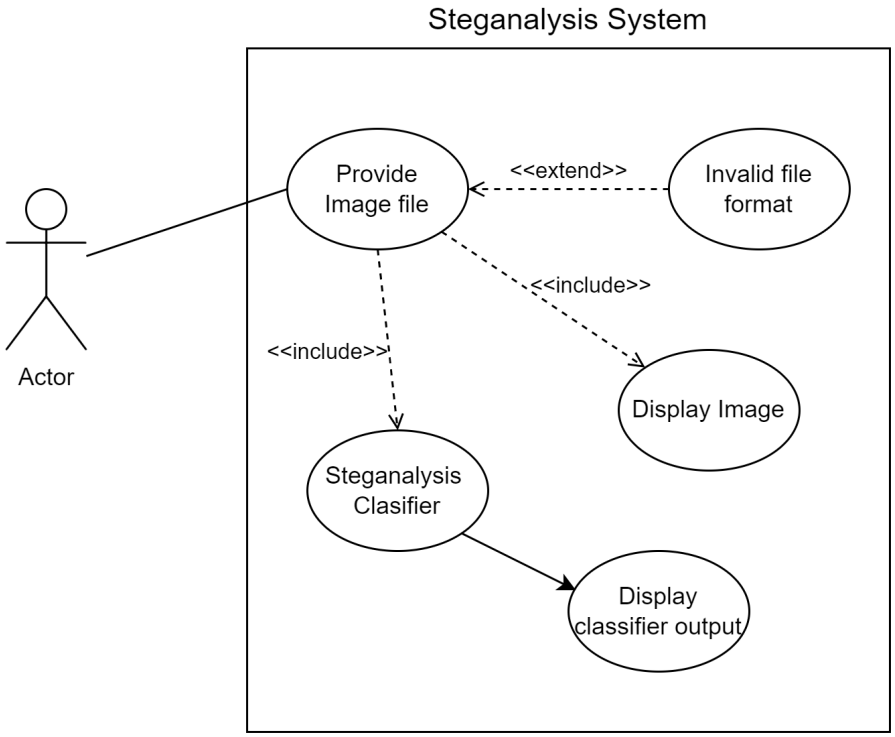


Figure 6.1: Use Case Diagram

Primary actor:

- User: Provides susceptible images to the system for analysis.

Use-Cases:

1. Provide Image file with Invalid Format (Extend):
    - This use case allows the user to provide an image file with JPEG extension to the system for analysis.
    - If the image file is invalid, the system will handle this error by notifying the user and prompting them to provide a valid image.
2. Display Image (Include):
    - This use case includes the functionality of displaying the image to the user.
    - After the user provides a valid image file, the system will display the image on the interface.
3. Steganalysis Classifier (Include):
    - This use case includes the functionality of performing steganalysis on the provided image.
    - After displaying the image, the system will apply the steganalysis classifier to analyze the image for hidden data.
4. Display Classifier Output:
    - The output of the steganalysis classifier is displayed to the user.

6.2 Sequence Diagram

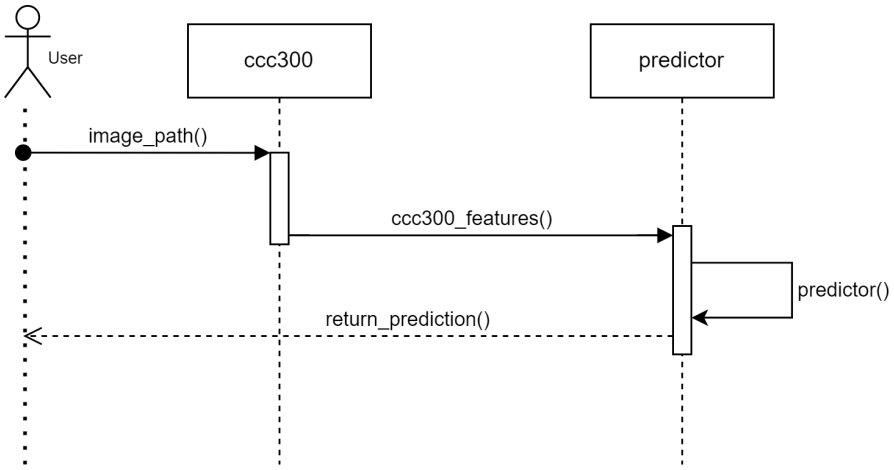


Figure 6.2: Sequence Diagram

Objects

1. User
2. CC-C300
3. Predictor

Steps

1. The User sends a message to the CC-C300 extractor to provide the path to the image.
2. The CC-C300 processes the image, extracts the features and sends a message "ccc300_features" to the Predictor to provide the features.
3. Finally, the Predictor sends the prediction result back to the User through the message "return_prediction()"

6.3 System Block Diagram

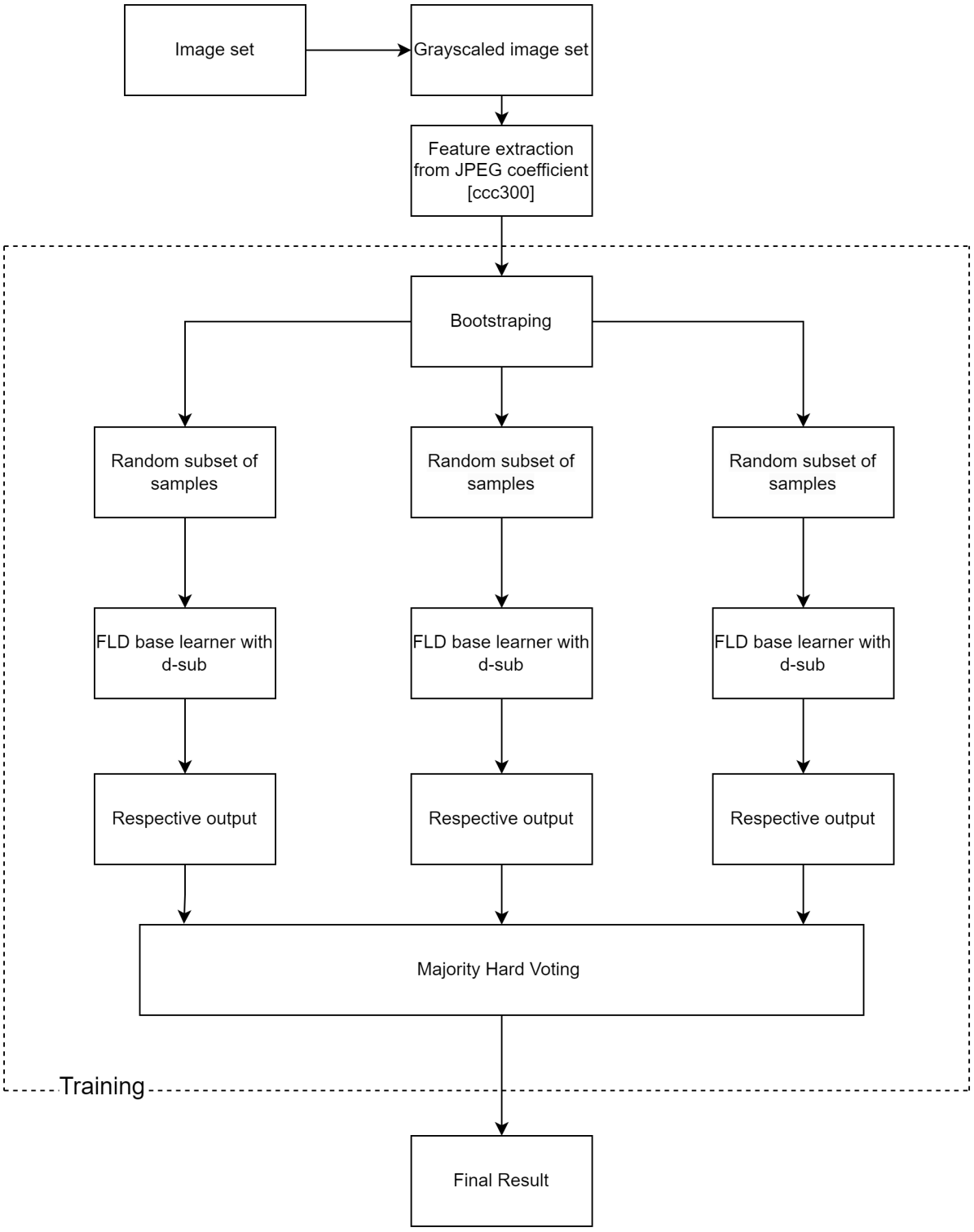


Figure 6.3: System Block Diagram

At first, starting with a set of images the procedure begins by converting the images to grayscale then extracting CC-C300 features from JPEG image. After that bootstrapping is utilized to create diverse training data by selecting random subsets. Now those bootstrapped feature will go through different Fisher Linear Discriminant base learner. By repeating this process across subsets, ensemble learning is achieved. Outputs from each base learner are sent for Voting. The final prediction is determined by combining the results using Hard Voting and is labeled as the “Majority Hard Voting ”. Then we will have our trained model that can generate a final result.

6.4 System Architecture

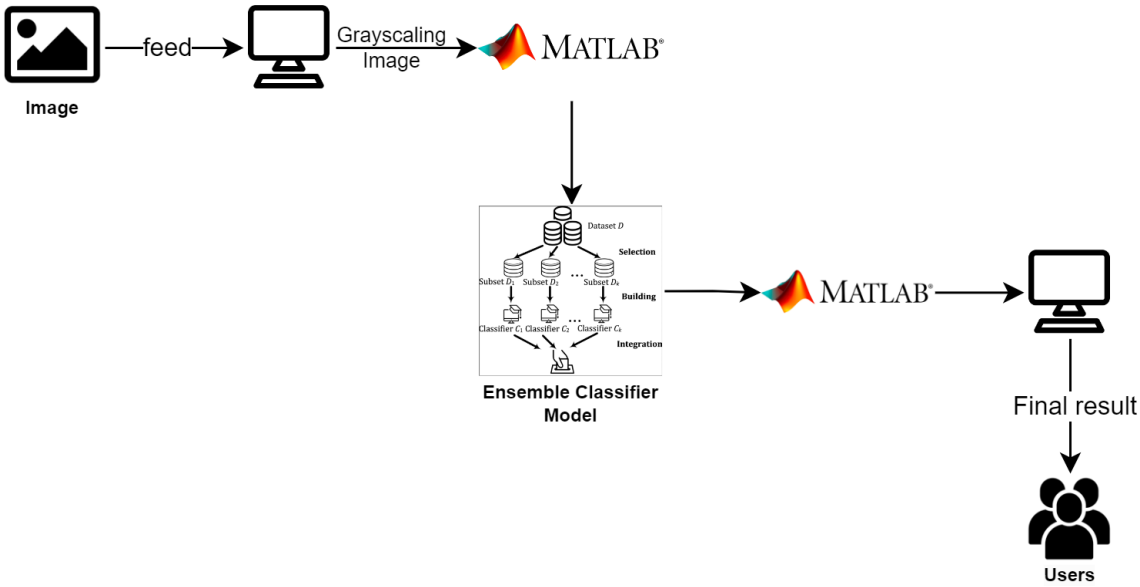


Figure 6.4: System Architecture

Chapter 7

Result and Discussion

7.1 Dataset Preparation

After the selection of our Dataset which is IStego100K [21], now we need to further process this dataset. It was not feasible to train the model using all data at once. So, Initially we splitted the dataset into different batches. Each batch consisting of 40,000 images (20000 cover images and 20000 stego images). We tried to train the model using this subset of datset but there was mermory Limitation problem. We then reduced the batch size to 30,000 (15,000 cover images and 15,000 stego images), but still faced the same issue. Finally, we set the batch size to 20,000 (10,000 cover and 10,000 stego), which allowed the training process to proceed successfully. To split the dataset, we utilized a Microsoft PowerShell script, which splitted the dataset into the desired subsets and stored them in respective subfolders.

7.2 Training the Ensemble Model

Before training the model, it was necessary to identify the most suitable features for our project. Therefore, we conducted several tests to determine the optimal feature set for training our model.

7.2.1 DCTR Feature

Initially, we used (DCTR) features with a feature dimensionality of 8000 for training our model. However, we faced several challenges: the feature extraction process was time-consuming, and the outcomes were unsatisfactory.

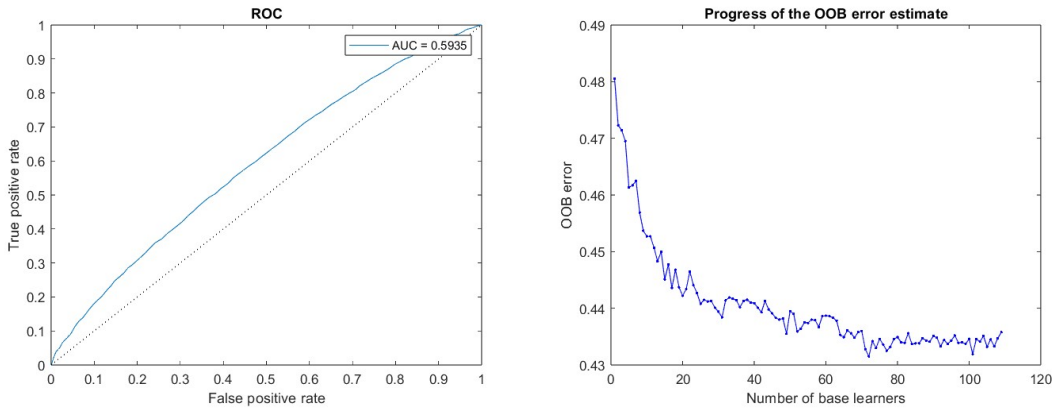


Figure 7.1: ROC curve and OOB vs No of base learner [DCTR]

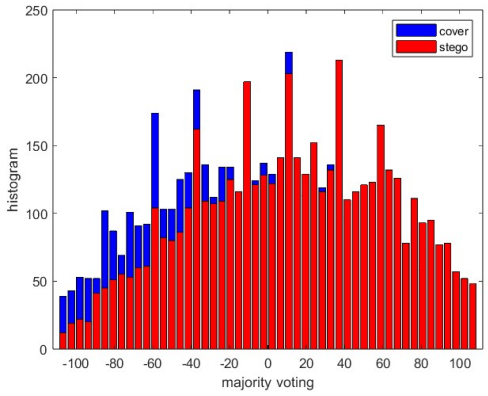


Figure 7.2: Histogram of votes [DCTR]

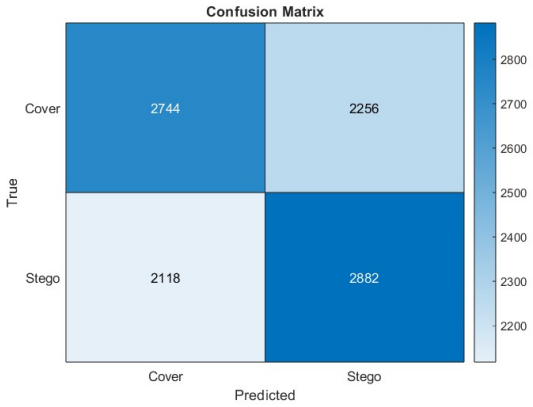


Table 7.1: Confusion Matrix [DCTR]

Using DCTR as feacture vector for trainig the model we got Area Under the Curve (AUC) of 0.59 which is quite low. We tried using different sub-set of data set but result was still same. Above graphs shows ROC curve, OOB error vs number of base learner, confusion matrix and histogram of votes.

7.2.2 CC-CN Feature

Later on, we came across cc-cN features, specifically cc-c300, which provided a higher dimensionality of 48600. These features proved to be more effective in capturing dependencies among individual cover elements. Due to the large dimensionality of our images, we used for Fisher's Linear Discriminant (FLD) as the base learner for our ensemble classifier. As a result, we observed improved performance compared to our earlier one.

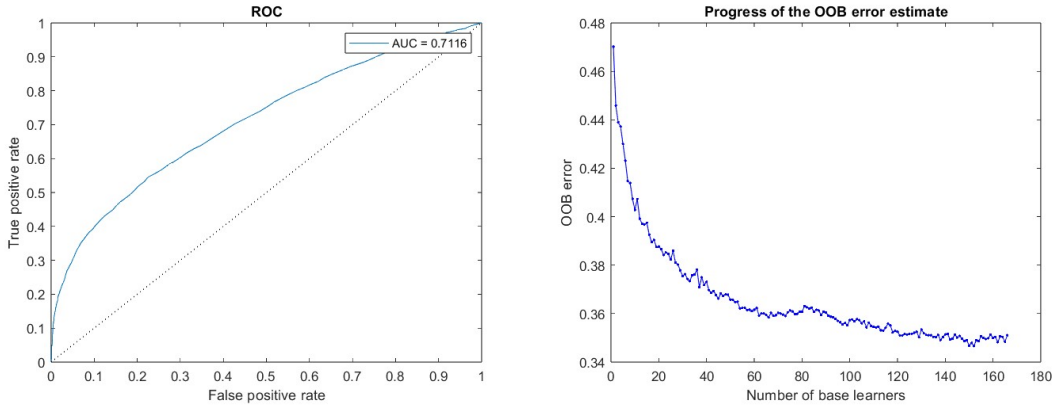


Figure 7.3: ROC and Search for subspace dimensionality [CC-C300]

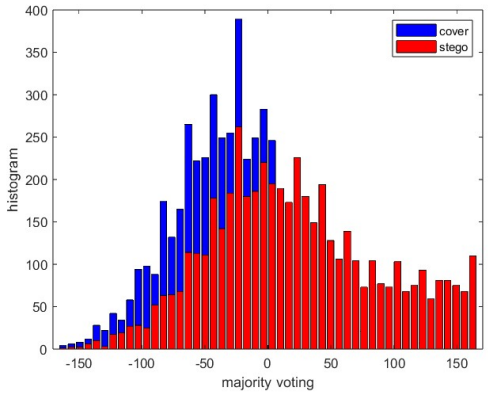


Figure 7.4: Histogram of votes [CC-C300]

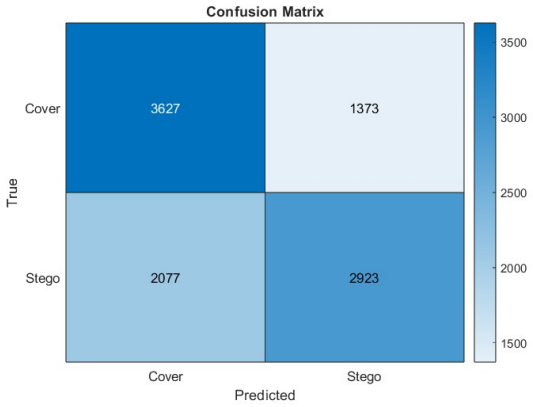


Table 7.2: Confusion Matrix [CC-C300]

Above graphs shows ROC curve, OOB error vs number of base learner, confusion matrix and histogram of votes.Using CC-C300 as feature vector for training the model we got Area Under the Curve (AUC) of 0.7116 which is comparatively higher than the DCTR model. Upon using different sub-set of datasets the result was still same. The OOB error was minimum when number of base learners was 160.

7.2.3 CC-CHEN Feature

We looked into another feature extraction method called cc-chen, which had 972 dimensions, to see if it was better than cc-c300. However, the results weren't as good as with cc-c300. So, we decided that cc-c300 was the best choice for our project.

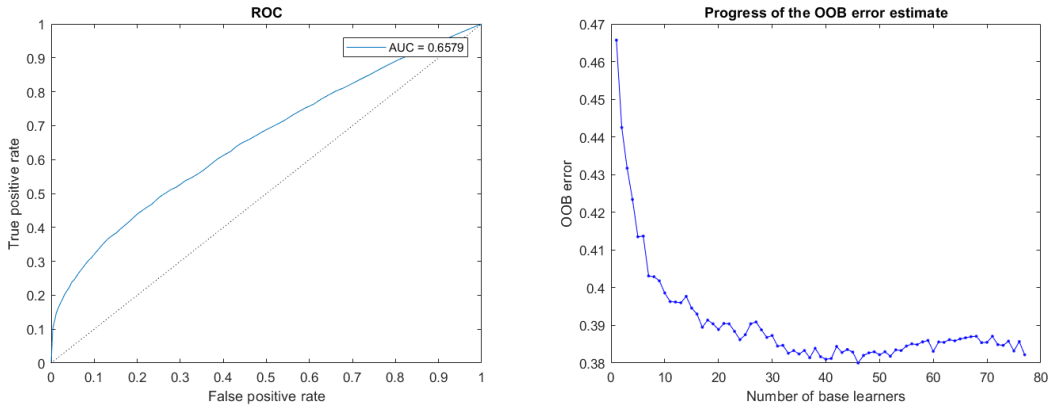


Figure 7.5: ROC and Search for subspace dimensionality [CC-CHEN]

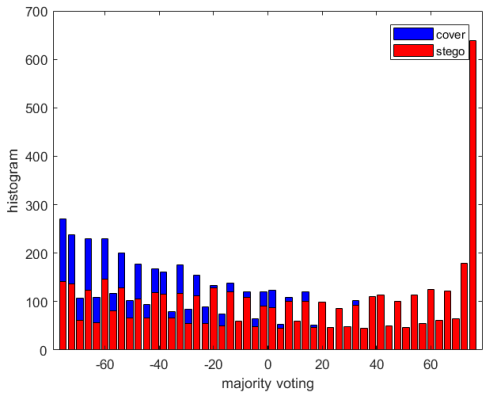


Figure 7.6: Histogram of votes [CC-CHEN]

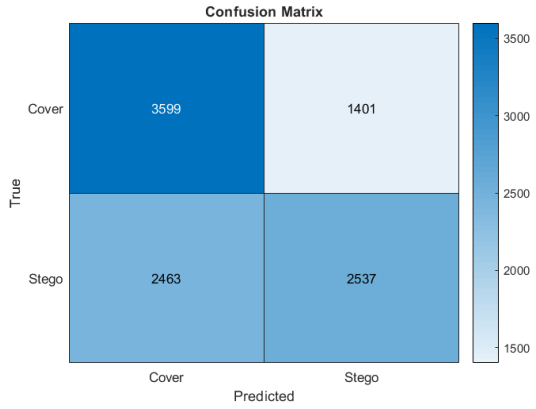


Table 7.3: Confusion Matirx [CC-CHEN]

Table below contiain summary of the performance of the model with different features.

Feature	Accuracy	Precision	Recall	F1 Score
DCTR	0.56	0.55	0.56	0.56
CC-CHEN	0.61	0.70	0.59	0.65
CC-C300	0.66	0.73	0.64	0.68

Table 7.4: Performance of the model with different features

7.3 Changing different parameters

After comparing with two other features we decied to choose CC-C300 feature. To get the optimal result we trained our model with different value of dsub and number of base learner. The results are shown below:

No of Base Learner	Subspace size	Accuracy	Precision	Recall	F1 Score
120	800	0.6947	0.79	0.60	0.68
100	1400	0.7045	0.64	0.62	0.63
160	2600	0.7116	0.73	0.64	0.69

Table 7.5: Performance of the model with different parameters

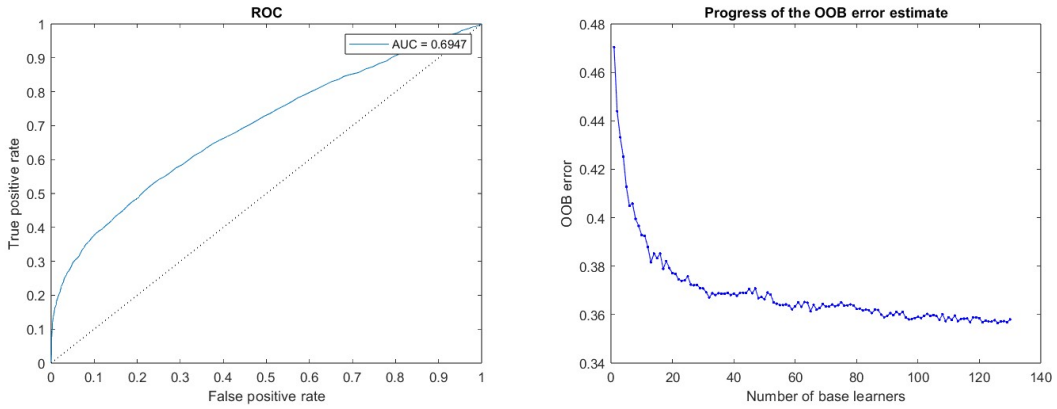


Figure 7.7: ROC curve and OOB vs No of base learner[dsub=800]

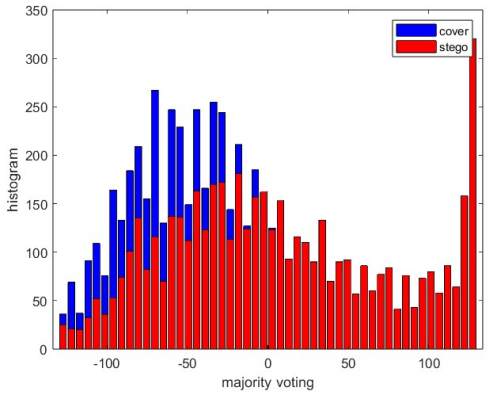


Figure 7.8: Histogram of votes [800]

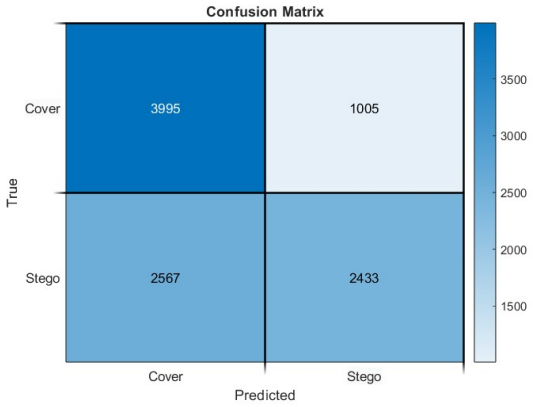


Table 7.6: Confusion Matrix [800]

Using 800 sub space dimensionality with CC-C300 feature vector we got Area Under the Curve (AUC) of 0.69. The ROC curve, OOB error vs number of base learner, confusion matrix and histogram of votes are shown in the above graphs. The OOB error was minimum when number of base learner was 120. In this process we made sub space dimensionality constant of 800 while number of base learner to automatically get selected when OOB error was minimum.

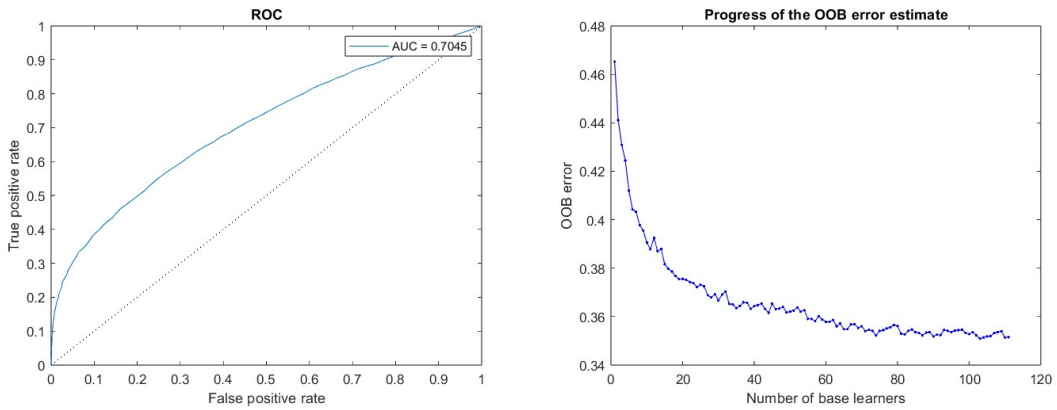


Figure 7.9: ROC curve and OOB vs No of base learner [dsub=1400]

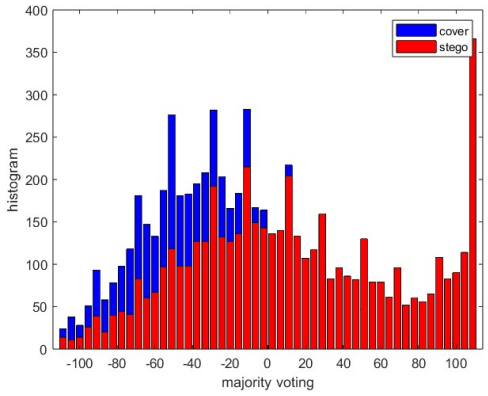


Figure 7.10: Histogram of votes [1400]

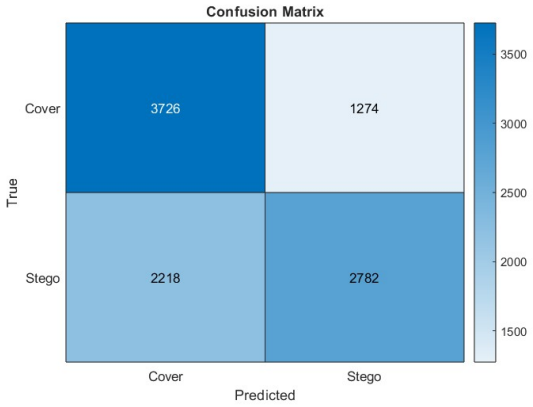


Table 7.7: Confusion Matrix [1400]

After that we changed sub space dimensionality to 1400, we examined a model's performance and got an Area Under the Curve (AUC) of 0.70. ROC curves, OOB error versus the number of base learners, a confusion matrix, and a vote histogram are shown above. The least amount of out-of-bag error (OOB) was obtained when 100 base learners were used.

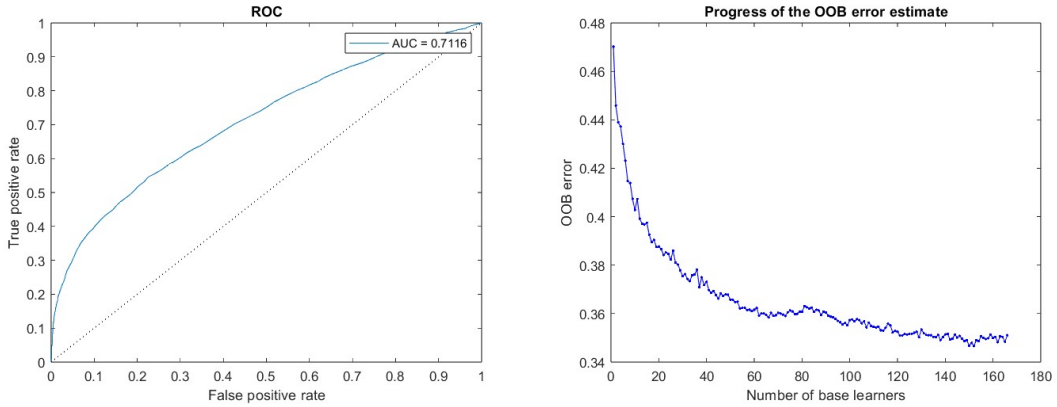


Figure 7.11: ROC curve and OOB vs No of base learner[dsub=2600]

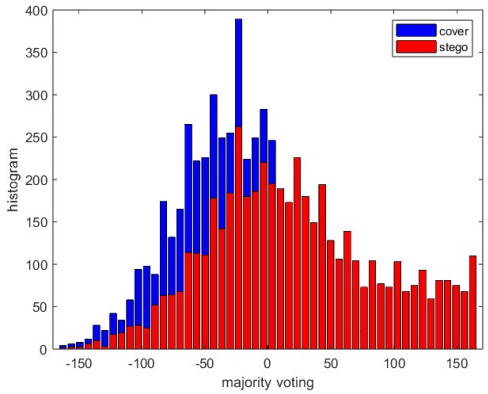


Figure 7.12: Histogram of votes [2600]

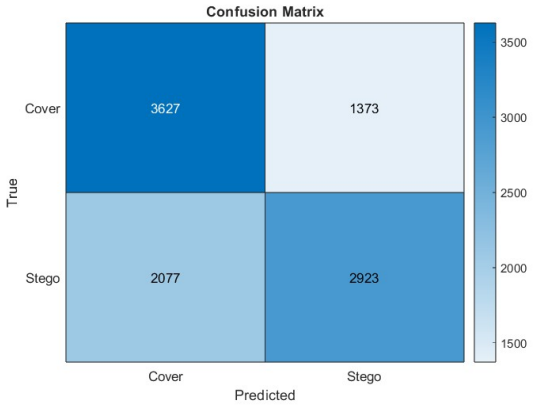


Table 7.8: Confusion Matrix [2600]

Then we check for 2600 subspaces, we examined a model's performance and got an Area Under the Curve (AUC) of 0.7116. This was better then all above results. . The OOB error was minimum when the number of base learners was 160. ROC curve, OOB error versus the number of base learners, a confusion matrix, and a vote histogram are shown above.

7.4 Discussion

The decision between MATLAB and Python as our primary platform was influenced by the availability of feature extraction tools. Since our feature extraction process was exclusive to MATLAB, we were inclined to use MATLAB. Although the functionality to call and use python engine inside MATLAB was available but there were limitations to their usage as there were version compatibility issues. For the functionality of using python in MATLAB we required matlabengine which had various bugs and they have various version compatibility issues as well.
There were also various procedure which had to be carefully followed in order to properly execute python libraries using matlabengine. Thus, although it was possible to call MATLAB functions from Python, limitations such as version compatibility and slower execution using MATLAB Engine prompted us to stick with MATLAB for feature extraction.
For the creation of the demos, there were functionalities to create web based demo using MATLAB as backend with help of FLASK but it required matlabengine which had various compatibility issues as mentioned before. So, we decided to use MATLAB's matlab app designer functionality for creating a demo app without the need of matlabengine.

7.5 Result

The following are the findings or results obtained from the selected model:

7.5.1 Time requirements

Table below shows the different time requirements for our project:

S.N	Average Time Required
Feature Extratoion	0.8 second per Image
Model Training with 20,000 images	10 minutes
Testing	30 second

Table 7.9: Time Requirements

7.5.2 ROC Curve

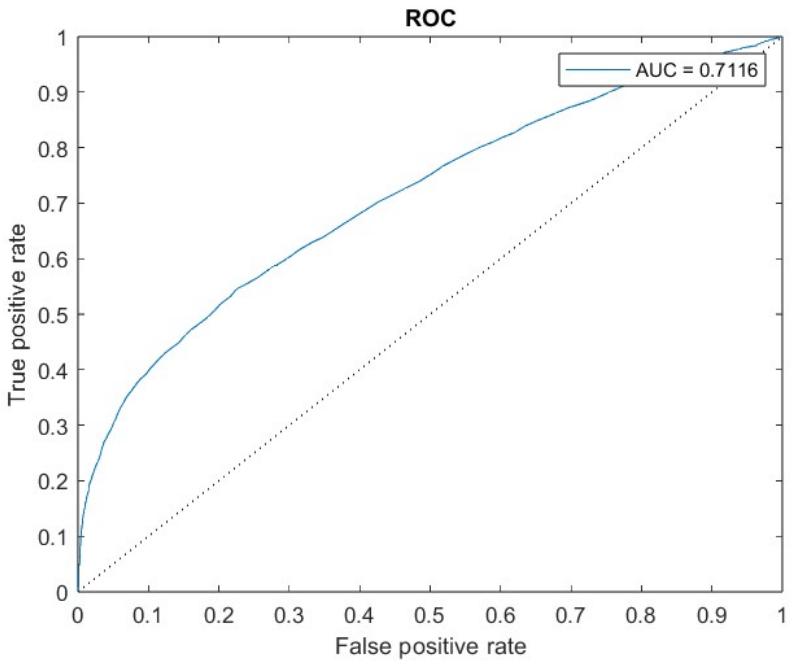


Figure 7.13: ROC Curve

The ROC curve is a graphical representation of the contrast between true positive rates and the false positive rate at various thresholds. It is often used as a proxy for the trade-off between the sensitivity and the specificity of the model. The AUC is the area under the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.
We obtained Area Under the Curve (AUC) of 0.7116, indicating a high level of classification between positive and negative instances

7.5.3 OOB versus Number of Base Learners

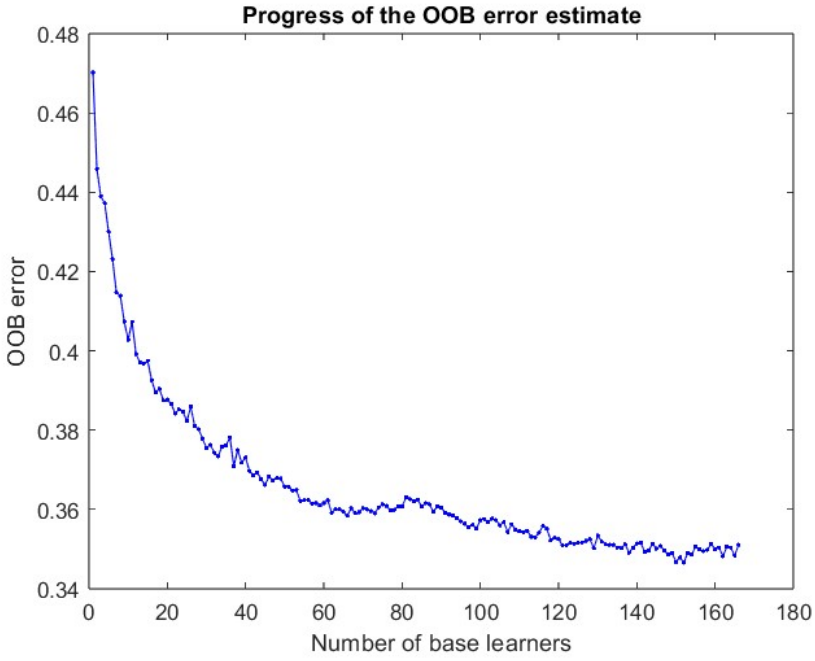


Figure 7.14: OOB vs Number of Base Learners

Above graph shows different values of OOB error for different number of base learners. It is observed that the OOB error decreases as the number of base learners increases. The OOB error is minimum when the number of base learners is 160.

7.5.4 Confusion Matrix

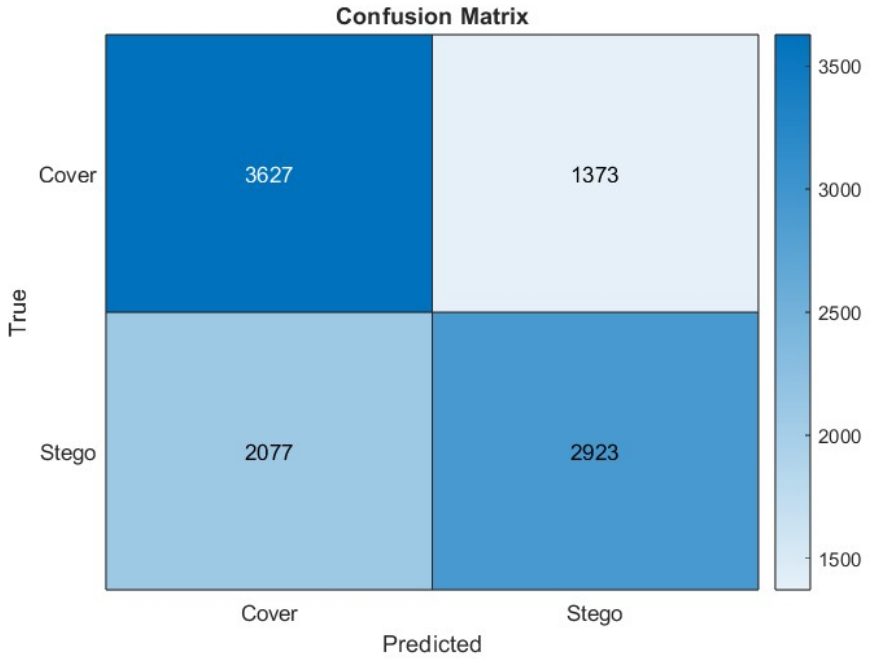


Table 7.10: Confusion Matrix

The model's classification performance is summarized through a confusion matrix, featuring 3627 true positives (correctly identified positives), 2923 false negatives (misclassified negatives), 2077 false positives (incorrectly labeled positives), and 1373 true negatives (correctly identified negatives). These metrics offer a concise yet comprehensive overview of the model's ability to distinguish between positive and negative cases, providing crucial insights for evaluating its effectiveness. The confusion matrix serves as a pivotal tool in assessing the model's strengths and weaknesses, contributing valuable information to the ongoing discourse on classification model refinement.

7.5.5 Histogram

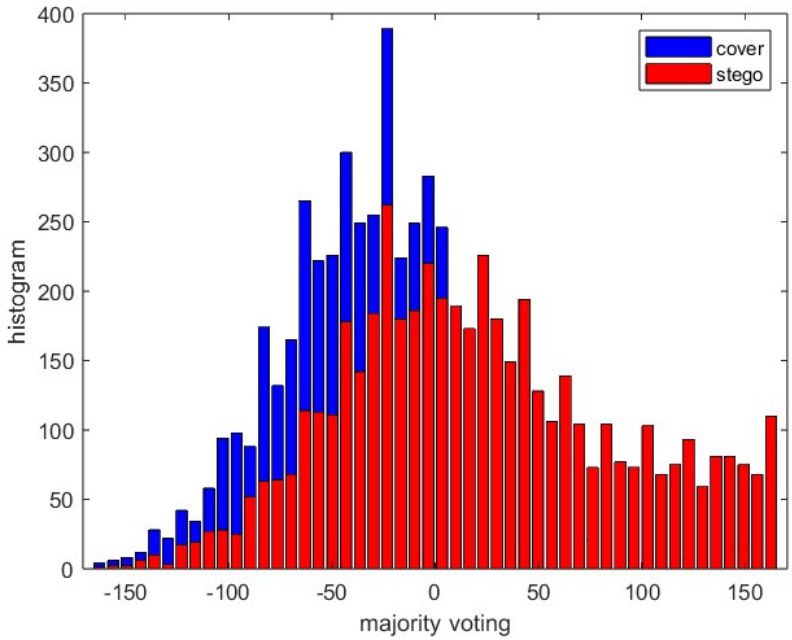


Figure 7.15: Histogram of votes

The histogram of votes is a graphical representation of the number of votes for each class. The histogram provides a visual representation of the model's classification performance, offering insights into the distribution of votes across the different classes. The histogram is a valuable tool for evaluating the model's classification performance, providing a comprehensive overview of the distribution of votes and the model's ability to distinguish between positive and negative cases.

7.6 Model Evaluation

7.6.1 Performance Metrices

7.6.1.1 Precision

$$Precision = \frac{TP}{TP + FP} = \frac{3627}{3627 + 1373} = 0.73$$

7.6.1.2 Accuracy

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{3627 + 1373}{3627 + 1373 + 2077 + 2923} = 0.66$$

7.6.1.3 Recall

$$Recall = \frac{TP}{TP + FN} = \frac{3627}{3627 + 2077} = 0.64$$

7.6.1.4 F1 Score

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} = 2 * \frac{0.73 * 0.64}{0.73 + 0.64} = 0.68$$

Chapter 8

Conclusion

The project's goals of using Ensemble Classifier to identify JPEG images altered through steganography were successfully met. Our major goal was to develop a powerful Ensemble Classifier that could identify hidden messages in JPEG images throughout the course of our image steganalysis research. We encountered numerous difficulties and gained a great deal of knowledge as we navigated the complexity of image processing and machine learning. Gaining the ability to understand and interpret research papers was one of the most essential lessons we took away. The field of steganalysis is very vast but there is very little progress seen on universal steganalysis , this project aims to further empower the research on universal steganalysis.
In conclusion, the project of steganalysis using ensemble classifier is completed successfully as it was able to properly detect steganographically modified images accurately

Chapter 9

Limitations and Future Enhancements

9.0.1 Limitations

These are some of the Limitations of our project that it currently faces:

1. The system could only be trained on a small dataset because we faced limitations in accessing sufficient computational resources.
2. The model has the ability to only detect steganography using three algorithms: nsF5, J-uniward, and UERD. It's optimized for detecting embedding rates ranging from 0.1 to 0.4 bits per pixel.
3. The system is confined to MATLAB, which might not be widely familiar among users.
4. The project is limited to JPEG images only. It does not support other image formats.

9.0.2 Future Enhancements

Considering our current limitations, we have identified the following areas for future enhancements:

1. Optimize model training by efficiently utilizing the entire dataset while accommodating limited computational resources.
2. Convert the existing code into Python format to enhance its versatility for implementation on various platforms.
3. Develop a Google Chrome extension that employs a model for detecting steganographically altered images on the internet.
4. Extend the model to support other image formats such as PNG, BMP, and TIFF.

Chapter 10

Bibliography

[1] Tayana Morkel, Jan HP Eloff, and Martin S Olivier. An overview of image steganography. In *ISSA*, volume 1, pages 1–11, 2005.

[2] Arooj Nissar and Ajaz Hussain Mir. Classification of steganalysis techniques: A study. *Digital Signal Processing*, 20(6):1758–1770, 2010.

[3] Syed Ali Khayam. The discrete cosine transform (dct): theory and application. *Michigan State University*, 114(1):31, 2003.

[4] Osama F AbdelWahab, Aziza I Hussein, Hesham FA Hamed, Hamdy M Kelash, Ashraf AM Khalaf, and Hanafy M Ali. Hiding data in images using steganography techniques with compression algorithms. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 17(3):1168–1175, 2019.

[5] Jessica Fridrich, Miroslav Goljan, and Dorin Hogea. Steganalysis of jpeg images: Breaking the f5 algorithm. In *Information Hiding: 5th International Workshop, IH 2002 Noordwijkerhout, The Netherlands, October 7-9, 2002 Revised Papers 5*, pages 310–323. Springer, 2003.

[6] Jessica Fridrich, Miroslav Goljan, and David Soukal. Efficient wet paper codes. In *International Workshop on Information Hiding*, pages 204–218. Springer, 2005.

[7] Jessica Fridrich, Petr Lisoněk, and David Soukal. On steganographic embedding efficiency. In *Information Hiding: 8th International Workshop, IH 2006, Alexandria, VA, USA, July 10-12, 2006. Revised Selcted Papers 8*, pages 282–296. Springer, 2007.

[8] NV Koshkina. J-uniward steganoanalysis. *Cybernetics and Systems Analysis*, 57(3):501–508, 2021.

[9] Junjun Gan, Jiufen Liu, Xiangyang Luo, Chunfang Yang, and Fenlin Liu. Reliable steganalysis of hugo steganography based on partially known plaintext. *Multimedia Tools and Applications*, 77(14):18007–18027, 2018.

[10] Xiangyang Luo, Xiaofeng Song, Xiaolong Li, Weiming Zhang, Jicang Lu, Chunfang Yang, and Fenlin Liu. Steganalysis of hugo steganography based on parameter recognition of syndrome-trellis-codes. *Multimedia Tools and Applications*, 75:13557–13583, 2016.

[11] Mikołaj Płachta, Marek Krzemień, Krzysztof Szczypiorski, and Artur Janicki. Detection of image steganography using deep learning and ensemble classifiers. *Electronics*, 11(10):1565, 2022.

[12] Jan Kodovsky, Jessica Fridrich, and Vojtech Holub. Ensemble classifiers for steganalysis of digital media. *IEEE Transactions on Information Forensics and Security*, 7, 04 2012.

[13] George Berg, Ian Davidson, Ming-Yuan Duan, and Goutam Paul. Searching for hidden messages: Automatic detection of steganography. In John Riedl and Randall W. Hill Jr., editors, *Proceedings of the Fifteenth Conference on Innovative Applications of Artificial Intelligence, August 12-14, 2003, Acapulco, Mexico*, pages 51–56. AAAI, 2003.

[14] Mark T Hogan, Neil J Hurley, Guénolé CM Silvestre, Félix Balado, and Kevin M Whelan. Ml detection of steganography. In *Security, Steganography, and Watermarking of Multimedia Contents VII*, volume 5681, pages 16–27. SPIE, 2005.

[15] Jan Kodovsky and Jessica Fridrich. Steganalysis in high dimensions: Fusing classifiers built on random subspaces. *Proceedings of SPIE - The International Society for Optical Engineering*, 02 2011.

[16] Jan Kodovsky, Jessica Fridrich, and Vojtech Holub. Ensemble classifiers for steganalysis of digital media. *IEEE Transactions on Information Forensics and Security*, 7, 04 2012.

[17] Arezoo Torkaman and Reza Safabakhsh. A fast and accurate steganalysis using ensemble classifiers. In *2013 8th Iranian Conference on Machine Vision and Image Processing (MVIP)*, pages 22–26, 2013.

[18] Jan Kodovský and Jessica Fridrich. Calibration revisited. In *Proceedings of the 11th ACM Workshop on Multimedia and Security*, MM&Sec '09, page 63–74, New York, NY, USA, 2009. Association for Computing Machinery.

[19] Yun Q. Shi, Chunhua Chen, and Wen Chen. A markov process based approach to effective attacking jpeg steganography. In Jan L. Camenisch, Christian S. Collberg, Neil F. Johnson, and Phil Sallee, editors, *Information Hiding*, pages 249–264, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[20] Tomáš Pevný and Jessica J. Fridrich. Merging markov and dct features for multi-class jpeg steganalysis. In *Electronic imaging*, 2007.

[21] Zhongliang Yang, Ke Wang, Sai Ma, Yongfeng Huang, Xiangui Kang, and Xianfeng Zhao. Istego100k: Large-scale image steganalysis dataset. In *International Workshop on Digital Watermarking*. Springer, 2019.

[22] Lior Rokach. Ensemble-based classifiers. *Artificial intelligence review*, 33:1–39, 2010.

[23] Rémi Cogranne, Tomas Denemark, and Jessica Fridrich. Theoretical model of the fld ensemble classifier based on hypothesis testing theory. In *2014 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 167–172. IEEE, 2014.

Appendix

A Demonstration

A.1 HomePage

Figure A-1: Home Page

A.2 Image Upload Section

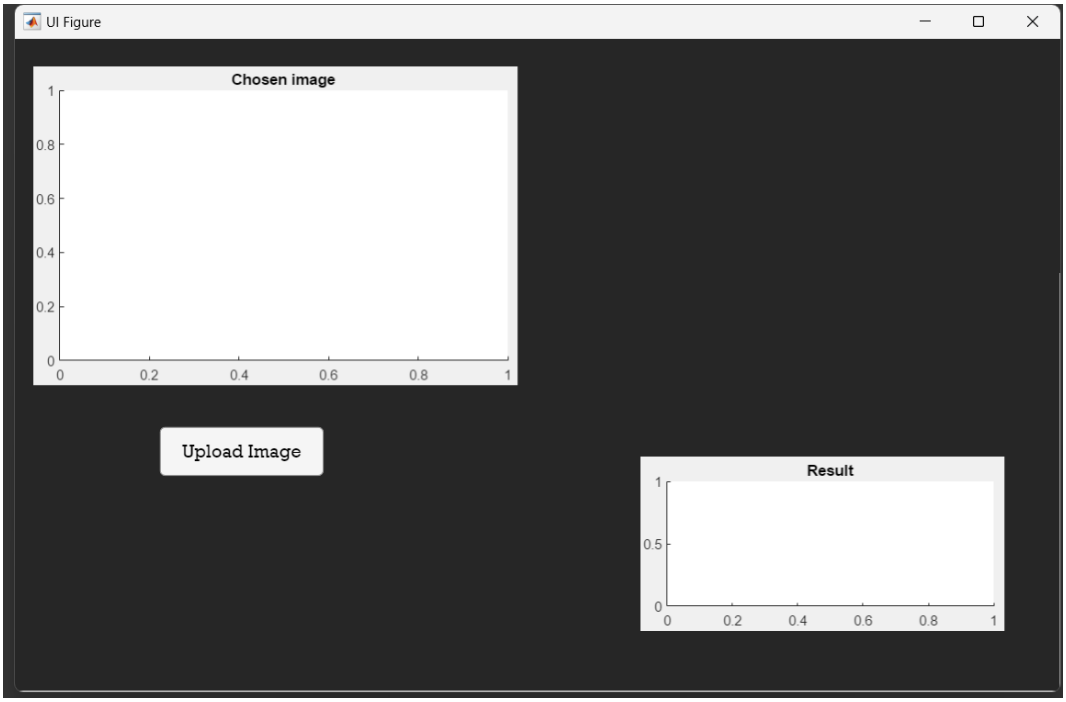


Figure A-2: Image Upload Section

A.3 Selecting an Image

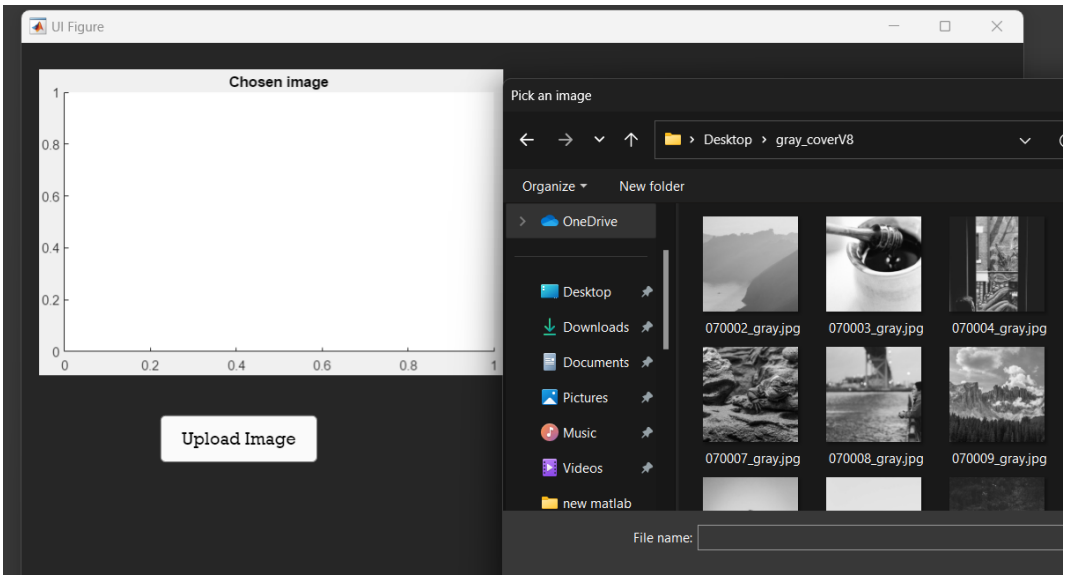


Figure A-3: Image Selection

A.4 Wrong Input Format

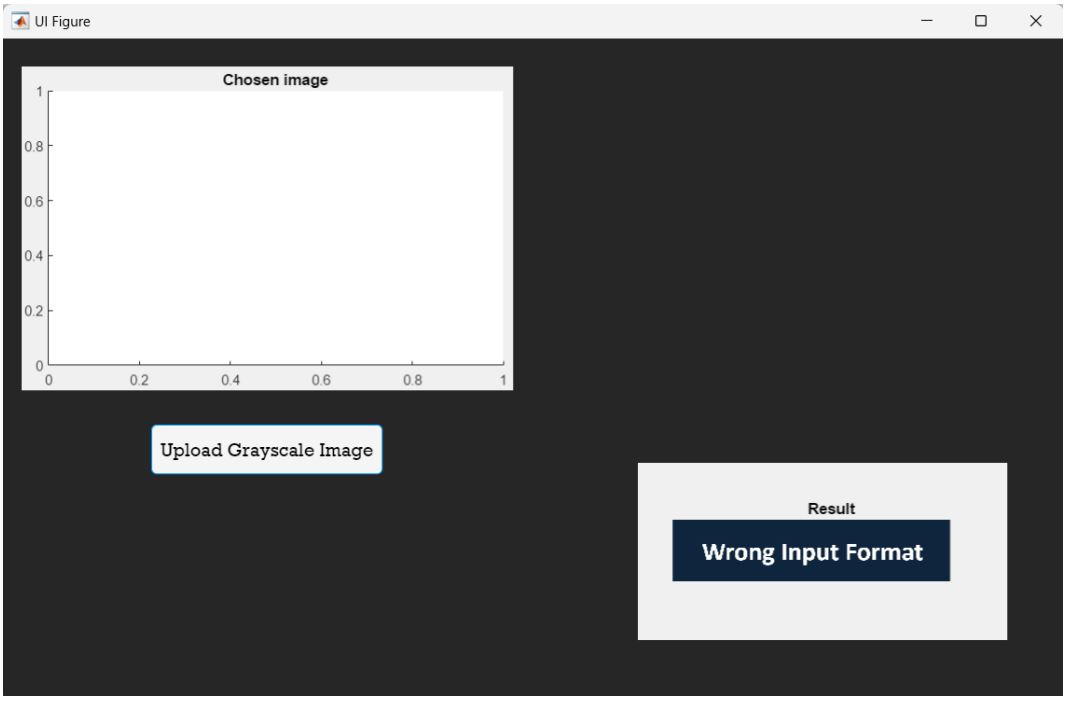


Figure A-4: wrong Input format Selection

A.5 Results

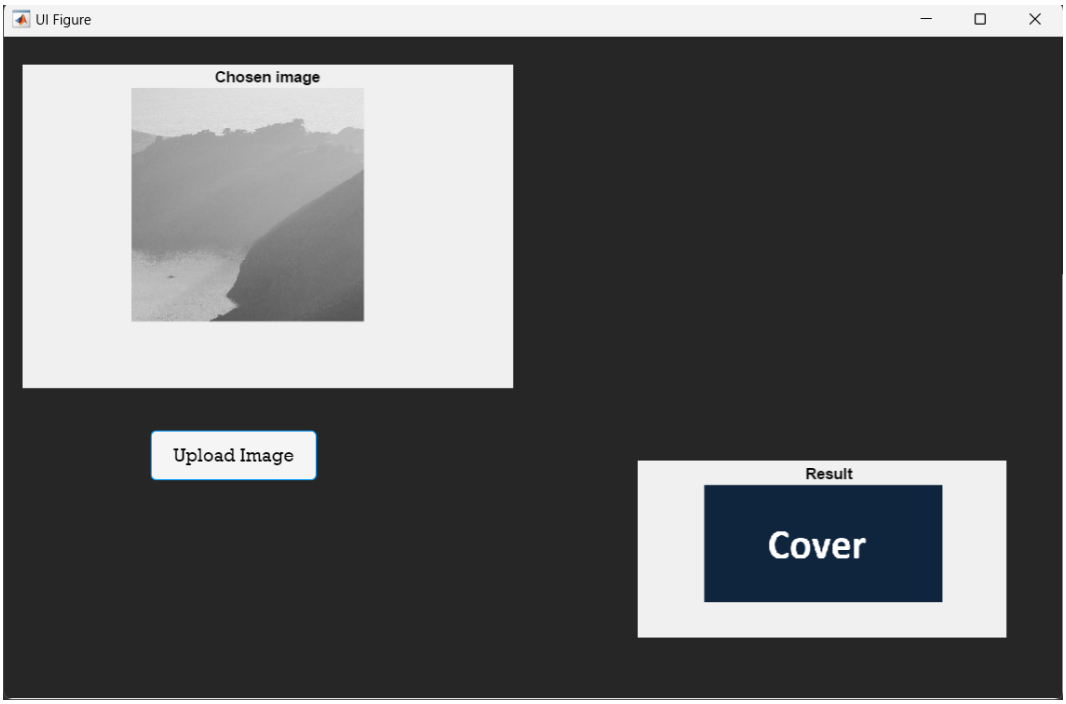


Figure A-5: Result

Figure A-6: Result

B Model Training

B.1 Splitting Dataset

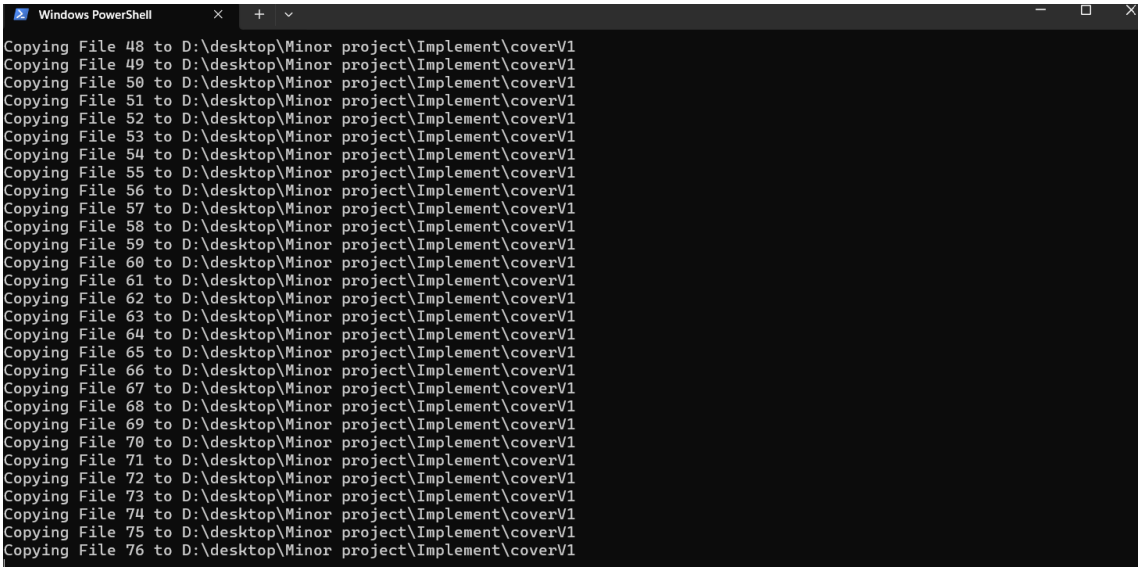


Figure A-7: Splitting Dataset using PowerShell Script

B.1.1 Result after Splitting

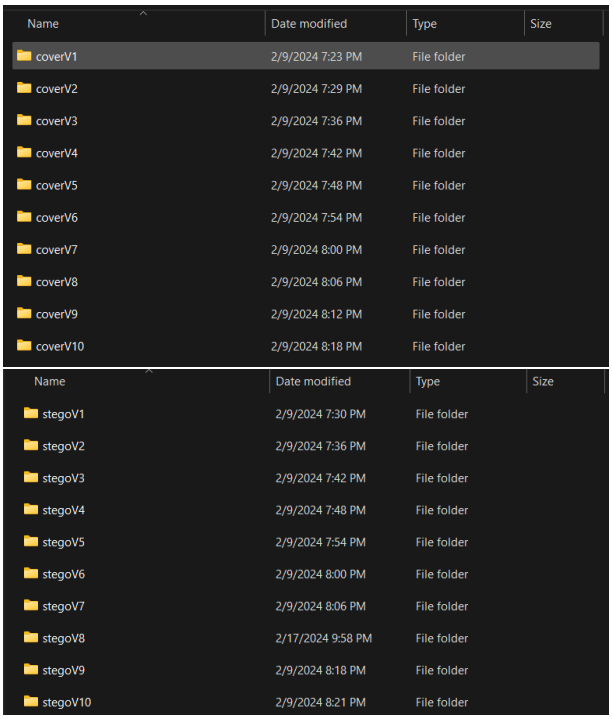


Figure A-8: Results after Splitting

B.2 Feature Extraction

```
 - processed in 0.80 seconds

Processing image: D:\projectMatlab\DCTR_matlab_v1.0\stegoV2 output python\010048_gray.jpg
 - processed in 0.61 seconds

Processing image: D:\projectMatlab\DCTR_matlab_v1.0\stegoV2 output python\010049_gray.jpg
 - processed in 0.68 seconds

Processing image: D:\projectMatlab\DCTR_matlab_v1.0\stegoV2 output python\010050_gray.jpg
 - processed in 0.74 seconds

Processing image: D:\projectMatlab\DCTR_matlab_v1.0\stegoV2 output python\010051_gray.jpg
 - processed in 0.72 seconds

Processing image: D:\projectMatlab\DCTR_matlab_v1.0\stegoV2 output python\010052_gray.jpg
 - processed in 0.89 seconds

Processing image: D:\projectMatlab\DCTR_matlab_v1.0\stegoV2 output python\010053_gray.jpg
 - processed in 0.60 seconds

Processing image: D:\projectMatlab\DCTR_matlab_v1.0\stegoV2 output python\010054_gray.jpg
 - processed in 0.89 seconds

Processing image: D:\projectMatlab\DCTR_matlab_v1.0\stegoV2 output python\010055_gray.jpg
```

Figure A-9: Feature Extraction Screenshot

B.3 Model Training

```
>> tutorial
# -------------------------
# Ensemble classification
#   - Training samples: 10000 (5000/5000)
#   - Feature-space dimensionality: 48600
#   - L : automatic (min 25, max 500, length 50, eps 0.00500)
#   - d_sub : automatic (Eoob tolerance 0.0200, step 200)
#   - Seed 1 (subspaces) : 538281461
#   - Seed 2 (bootstrap) : 63638808
# -------------------------
 - d_sub 200 : OOB 0.3956 : L 94 : T 2.8 sec
 - d_sub 400 : OOB 0.3732 : L 88 : T 3.3 sec
 - d_sub 600 : OOB 0.3663 : L 123 : T 6.9 sec
 - d_sub 800 : OOB 0.3679 : L 96 : T 7.4 sec
 - d_sub 1000 : OOB 0.3625 : L 106 : T 11.1 sec
 - d_sub 1200 : OOB 0.3608 : L 74 : T 10.2 sec
 - d_sub 1400 : OOB 0.3516 : L 123 : T 21.8 sec
 - d_sub 1600 : OOB 0.3497 : L 136 : T 30.5 sec
 - d_sub 1800 : OOB 0.3538 : L 151 : T 45.2 sec
# -------------------------
optimal d_sub 1600 : OOB 0.3497 : L 136 : T 139.6 sec
```

Figure A-10: Model Training Screenshot