Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Software Department

STEGANALYSIS OF CONTENT-ADAPTIVE IMAGE STEGANOGRAPHY BASED ON CONVOLUTION NEURAL NETWORK AND SUPPORT VECTOR MACHINE

A Dissertation

Submitted to the Council of the College of Information Technology, University of Babylon in Partial Fulfillment of the Requirements for the Doctor of Philosophy Degree in Information Technology/Software

By

Saeed Mohammed Hashim Yaseen

Supervised by

Prof. Dr. Dhia Abdulhussein Jumaa Alzubaydi

2022 A.D. **1443 A.H.**

بسم الله الرحمن الرحيم

اللهُ نُورُ السَّمَاوَاتِ وَالْأَرْضِ مَثَلُ نُورِهِ كَمِشْكَاةٍ فِيهَا مِصْبَاحٌ الْمِصْبَاحُ فِي زُجَاجَةٍ الزُّجَاجَةُ كَأَنَّهَا كَوْكَبٌ دُرِّيٌّ يُوقَدُ مِن شَجَرَةٍ مُّبَارَكَةٍ زَيْتُونِةٍ لَّا شَرْقِيَّةٍ وَلَا غَرْبِيَّةٍ يَكَادُ زَيْتُهَا يُضِيءُ وَلَوْ لَمْ تَمْسَسْهُ نَارٌ نُورٌ عَلَى نُورٍ يَهْدِي اللهُ لِنُورِهِ مَن يَشَاء وَيَضْرِبُ اللهُ الْأَمْثَالَ لِلنَّاسِ وَاللهُ بِكُلِّ شَيْءٍ عَلِيمٌ (٣٥)

صدق الله العلي العظيم

سورة النور– اية ٣٥

Declaration

I hereby declare that this dissertation, submitted to the University of Babylon as fulfillment of requirements for the degree of doctor of Philosophy in Information Technology\ Software has not been submitted as an exercise for a similar degree at any other university. I also certify that the work described here is entirely my own.

Signature:

Name: Saeed Mohammed Hashim Yaseen

Date: / 3 / 2022

Supervisor Certification

I certify that the dissertation entitled **(Steganalysis of Content-Adaptive Image Steganography Based on Convolution Neural Network and Support Vector Machine)** was prepared under my supervision at the department of Software / College of Information Technology/ University of Babylon as partial fulfillment of the requirements of the degree of Doctor of Philosophy in Information Technology-Software.

Signature:
Supervisor Name: Prof. Dr. Dhia Abdulhussein Jumaa Alzubaydi
Date: / 3 / 2022

Head of the Department Certification

In view of the available recommendations, I forward the dissertation entitled **"Steganalysis of Content-Adaptive Image Steganography Based on Convolution Neural Network and Support Vector Machine "** for debate by the examination committee.

Signature:
Asst. Prof. Dr. Ahmed Saleam
Head of Software Department
Date: / 3 / 2022

Certification of the Examination Committee

We, the undersigned, certify that (Saeed Mohammed Hashim Yaseen) candidate for the degree of Doctor of Philosophy in Information Technology-Software, has presented his dissertation of the following title (**Steganalysis of Content-Adaptive Image Steganography Based on Convolution Neural Network and Support Vector Machine**) as it appears on the title page and front cover of the dissertation that the said dissertation is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on: (3 / 3 /2022).

Signature:
Name: Dr. Israa Hadi Ali
Title: Professor
Date: / 3 / 2022
(**Chairman**)

Signature:
Name: Dr. Wesam Sameer Bhaya
Title: Professor
Date: / 3 / 2022
(**Member**)

Signature:
Name: Dr. Alaa Kadhim Farhan
Title: Professor
Date: / 3 / 2022
(**Member**)

Signature:
Name: Dr. Methaq Talib Gaata
Title: Assistant Professor
Date: / 3 / 2022
(**Member**)

Signature:
Name: Dr. Wafaa Mohammed Saeed
Title: Assistant Professor
Date: / 3 / 2022
(**Member**)

Signature:
Name: Dr. Dhia Abdulhussein Jumaa
Title: Professor
Date: / 3 / 2022
(**Member and Supervisor**)

Approved by the Dean of the College of Information Technology, University of Babylon.

Signature:
Name: Dr. Hussein Atiya Lafta
Title: Professor
Date: / / 2022
(Dean of Collage of Information Technology)

Dedication

To the sake of Allah, my Creator, and my Master,

to my parents and my kids,

to my wife, my pillar of support. Thanks for holding me when I fell and helping me rise above and beyond,

to my brothers and sisters, who supported me until the completion of this research,

and to my friends, colleagues, and relatives,

I dedicate this work.

Acknowledgement

First and foremost, I would like to thank my God, Allah Almighty, for giving me endless graces. My deep sense of gratitude goes to the beacon of science, to the master of creatures, to the greatest Prophet, Mohammed (Peace be upon Him and His Family).

I take this opportunity to express my sincere gratitude and greatest appreciation to my supervisor **Prof. Dr. Dhia Abdul-Hussein Jumaa** for his continuous support for my Ph.D. study, his patience, motivation, enthusiasm, and immense knowledge. The words are inadequate and I can't say thank you enough for his tremendous support and help. I feel motivated and encouraged every time I attend his meeting. His tireless guidance has helped me immensely in researching and writing this dissertation.

Also, I would like to show my gratitude to who weaved my happiness from strings woven from her heart to my dear Mother. My thanks and appreciations also go to my family, for their encouragement, support and patience.

I wish to express my love and gratitude to my beloved wife for her understanding and endless love through the duration of my study.

Finally, I would also like to express my thanks and gratitude to all those who contributed to making this dissertation possible, and foremost among them all the teaching and staff members at the college of Information Technology at the University of Babylon.

Abstract

Steganography and steganalysis are two contradictory techniques. Steganography is a technique to hide secrete information into multimedia content in order to make it undetectable. The technique that identify whether there is secret information hidden in images is known as image steganalysis. Since content-adaptive image steganographic methods adaptively integrate knowledge into regions with distortion-based rich textures, state-of-the-art Image steganalysis approaches with the development of steganography technology cannot discriminate between cover and stego images, or detected with lower accuracy performance, and this makes it challenging to extract effective features.

Due to the impressive performance of Convolution Neural Network (CNN) in the field of image processing, a growing number of researchers have focused on developing steganalysis methods based on CNN. However, this dissertation aims to develop a system that based on CNN as a feature extraction and select the most informative features for steganalysis of content-adaptive image steganography.

After preparing the steganographic datasets, the proposed method applies Noise Residuals Filter to these datasets to suppress the image contents and exposing the stego noise component residuals, then applying the proposed CNN-based method to extract the image features. After that, use a Binary Particle Swarm Optimization Algorithm to select the most informative features, and finally, train the Support Vector Machine classifier using the selected training set and using the same classifier to discriminate between the normal images and images containing the secrete information for testing sets as a new unpredicted images.

Five content-adaptive steganographic methods are used to evaluate the results of the proposed method. The detection accuracy results ranged from 69.80% to 85.96%, from 71.00% to 87.53%, from 71.22% to 85.12%, from 60.77% to 88.16%, and from 65.39% to 87.04% for the HILL, HUGO, MiPOD, S-UNIWARD and WOW steganographic methods, respectively.

Publications Associated with this Dissertation

Some of the works presented in this dissertation have been published as listed below.

1. S. M. Hashim and D. A. Alzubaydi, "Identify the Presence of Hidden Information Based on Lower Coefficients Value of 2DHWT Sub-bands," 2021 IEEE 7th International Engineering Conference “Research & Innovation amid Global Pandemic" (IEC), 2021, pp. 156-161, doi: 10.1109/IEC52205.2021.9476121.
2. S. M. Hashim and D. A. Alzubaydi, "Steganalysis of Adaptive Image Steganography using Convolution Neural Network and Blocks Selection," 2021 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT), 2021, pp. 162-167, doi: 10.1109/COMNETSAT53002.2021.9530784.

Table of Contents

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

Abbreviation	Description
ABS	Absolute
ACA	Ant Colony Algorithm
AdaGrad	Adaptive Gradient
Adam	Adaptive Moment Estimation
APSO	Adaptive inertia-weight based Particle Swarm Optimization
BOSSbase	Break Our Steganographic System base
bpp	bit per pixel
BPSO	Binary Particle Swarm Optimization
BPSO_LR	Binary Particle Swarm Optimization_Logistic Regression
CCJRM	Cartesian-Calibrated JPEG Rich Model
CNN	Convolution Neural Network
CSA	Clone Selection Algorithm
DAMs	Diverse Activation Modules
DFMs	Diverse Filter Modules
DFSE-Net	Diverse Filters and Squeeze-and-Excitation Network
DNNs	Deep Neural Networks
DT	Decision Tree
ELU	Exponential Linear Unit
FM	Feature Map
FS	Feature Selection
GAV	Global Average

Grad-CAM	Gradient-weighted Class Activation Mapping
GWA	Grey Wolf Algorithm
HILL	High-pass, Low-pass, and Low-pass
HUGO	Highly Undetectable Stego Bounding Distortion
IAS-CNN	Image adaptive steganalysis via Convolution Neural Network
LBP	Local Binary Pattern
LR	Logistic Regression
LReLU	Leaky Rectified Linear Unit
LSBs	Least Significant Bits
LSTM	Long Short Term Memory
MiPOD	Minimizing the Power of Optimal Detector
MOOPs	Multi-Objective Optimization Problems
NRF	Noise Residual Filter
NRs	Noise Residuals
PGM	Portable Gray Map
PSO	Particle Swarm Optimization
PSRM	Projection Spatial Rich Model
RF	Random Forest
ReLU	Rectified Linear Unit
ReSt-Net	ReLU, Sigmoid, and TanH Network
RMSProp	Root Mean Squared Propagation
RNNs	Recurrent Neural Networks
SNR	Signal-to-Noise Ratio
SPAM	Subtractive Pixel Adjacency Matrix

SRM	Spatial Rich Model
SRNet	Steganalysis Residual Network
STCs	Syndrome Trellis Codes
S-UNIWARD	Spatial-Universal Wavelet Relative Distortion
SVM	Support Vector Machine
TanH	Hyperbolic Tangent
TLBP	Threshold Local Binary Pattern
TLU	Truncated Linear Unit
WOA	Whale Optimization Algorithm
WOW	Wavelet Obtained Weights

CHAPTER ONE

GENERAL INTRODUCTION

CHAPTER ONE
GENERAL INTRODUCTION

1.1 Overview

Steganography is a procedure of making the presence of hidden information in a host signal cannot be detected, such as text, video, and images. The hidden information that are embedded within a larger cover in such a way that the observer cannot identify the existence of this information [1, 2].

A steganographic application was described in 1980 when British Prime Minister Margaret Thatcher programmed word processors to encode the identities in the word space to tracing disloyal ministers responsible for the leaks of the cabinet's documents. [3, 4]

Today, Internet/storage devices (communication channels) provide an open area for exchanging information because they are a simple and ideal carrier. The data might be in the form of images, videos, or audio. Such medium can be utilized to hide secret information that is subsequently transmitted over the communication channel [5]. However, it might be harmful to communicate the secret information publicly, as privacy and secrecy are not guaranteed [6]. In such cases, it is necessary to send and receive information to and from certain individuals secretly. The communication via such a channel is termed secret communication.

There are a variety of steganographic tools available for hiding information within another cover file [7]. These cover files are usually digital multimedia files that are the optimal choice for steganography, as they are not suspicious (given the vast quantity of images on the Internet) and simple to process.

The techniques used to detect the existence of such non-traviliar distortions are termed steganalysis to discriminate the original cover media from that containing a secrete message. Because a steganalytic must assess if a particular digital media is "stego" or not, it is considered a hypothesis-testing issue. The most commonly used tools in the steganalysis communities are regularized linear classifiers [8], Ensemble classifiers [9], and Convolutional Neural Networks (CNNs) [10].

The development of steganalysis has forced the research community to develop new steganographic approaches to cope with the new and well-designed steganalytic models. To satisfy this purpose, the researchers propose increasingly complicated and sophisticated distortion functions to get a better cost-map [11, 12, 13]. In contrast, steganalysis should discover new image descriptors or models, which might cope with these newly updated steganographic techniques since the area of steganalysis is now challenged by this new generation of steganographic Algorithms.

The endless struggle between steganography and steganalysis creates an environment that stimulates itself and benefits both fields. In this context, it is necessary to develop a more suitable and effective set of methods.

1.2 Related Works

In this section, an overview of related works is provided, simultaneously trying to place the proposed method into its historical context.

In 2016, Xu et al. [14, 15] proposed a method of Convolution Neural Network (CNN) for steganalysis that including absolute (ABS)

activation and Batch Normalization (BN) layers. The ABS layer is used to enhance statistical modeling in subsequent layers, and BN avoids falling the network training being trapped in a bad area around local minima and optimizing feature map (FM) scales and biases. Moreover, used the Hyperbolic Tangent (*TanH*) activation function [16] to prevent over-fitting. The experiments of the proposed method satisfy better performance when compared with other steganalysis methods. It detection accuracy reached 81.29% using S-UNIWARD steganographic algorithm.

In 2017, Li et al. [17] proposed a new method to extract the features for image steganalysis based on "Threshold Local Binary Pattern (TLBP) Operator," which is performed on residual images. The high-order derivative filters generate the residual images to capture the stego artifacts. Then, second-order co-occurrence matrix features are extracted. Finally, the ensemble classifier is trained as a detector. The results had proved that the TLBP is competitive to SRM [18]. HILL, MiPOD and S-UNIWARD steganographic methods are used for evaluation. The results ranged from 58.94% to 77.05, from 59.25% to 77.29%, and from 59.38% to 81.07 detection accuracies, respectively.

In 2018, Li et al. [19] proposed ReSt-Net, a steganalysis technique based on multiple activation modules and parallel subnet-based CNN. Their design consists of diverse activation modules (DAMs) that activate the convolution outputs in different ways before aggregating their results for subsequent layers. The network used more sub-nets with fewer filters rather than increasing the number of filters for pre-processing layers. To speed up the training process, pre-trained the subnets individually. The evaluation results performed using S-UNIWARD, HILL, and CMD-HILL

and their results ranges from 65.67% to 85.44%, from 62.38% to 81.66%, and from 58.92% to 79.16%, respectively.

In 2018, Zhou [20] proposed a steganalysis approach that makes two new selection-channel information. First, use a mapping function to enhance the information, and this improved information is then combined with co-occurrence modeling to identify pixels with a high embedding modification probability. In this approach, the selection-channel information is further exploited, and the resulted features are influenced more by pixels with a significant embedding modification. Experiments demonstrate that the αSRM outperforms other steganalysis features. The evaluation results performed using S-UNIWARD, HILL, MiPOD, CMD-S-UNIWARD, CMD-HILL, and CMD-MiPOD. The results ranges from 63.75% to 85.04%, from 64.67% to 83.36%, and from 59.94% to 81.11%, from 63.22% to 80.44%, from 63.37% to 80.39%, and from 57.47% to 76.00%, respectively. This approach used 10% and 50% bpp as the lowest and highest embedding rates.

In 2019, Lu Jicang et al. [21] proposed F-opt, an enhanced framework for steganalysis of content-adaptive image steganography based on FS and pre-classification approaches. The authors used the k-means method image dataset to extract images with differing textures and complexity. The optimum features for each cluster are then determined for the final choice to enhance the overall performance. This method used only 10% and 30% bpp as embedding rates in the evaluation process. The satisfied results include: 63.36% and 78.05%, 58.08% and 70.21%, and 59.1% and 72.96% for S-UNIWARD, MiPOD and HUGO steganographic methods, respectively.

In 2019, Liu et al. [22] proposed a new steganalysis methodology that using a nature-inspired FS technique depending on the binary bat Algorithm. This approach identifies the most compelling feature subset from the features that extracted by Subtractive Pixel Adjacency Matrix (SPAM) [23] method to improve detection accuracy. Experiment results demonstrated that the proposed approach enhances detection while reducing redundant features. This approach evaluated using HILL, WOW, and HUGO steganographic methods at 40% bpp embedding rate. The results satisfied 64.11%, 68.08% and 64.07% detection accuracies, respectively.

In 2019, Huang, Min [24] proposed a CNN architecture that utilizes 30 trainable high-pass filters in the architecture's first convolutional layer to make it self-learning. Some constraints, such as symmetry and keeping the sum of the weights at zero, have been enforced on these filters to make them operate as high-pass filters throughout the training stage. The experiments illustrated that the proposed architecture outperforms other handcrafted-based steganalysis methods. This approach evaluated using WOW, MiPOD and S-UNIWARD steganographic methods at 20%, 30%, and 40% bpp embedding rate. The satisfied results ranged from 61.2% to 72.6% , from 62.3% to 75.1%, and from 62.1% to 73.1% detection accuracies, respectively.

In 2020, Zhujun et al. [25] proposed an approach called IAS-CNN. As a pre-processing stage, this technique uses self-learning to improve the filter. Create the filter manually at first to initiate the pre-processing, then include it into the CNN learning process. Additionally, this technique incorporates selection channel knowledge in image pre-processing to enhance the residuals and start the network with high payload data

training parameters to improve network performance. IAS-CNN is also utilizes fewer resources and processing data more quickly. The evaluation results performed using HUGO, S-UNIWARD, and WOW steganographic methods at 20% and 40% bpp embedding rates. The results ranges from 70.45% to 78.35%, from 62.4% to 75.05%, and from 68.15% to 80.75%, respectively.

In 2020, Xiang et al. [26] proposed a steganalysis technique based on CNNs, which includes two contributions: First, it offers a new structure of convolutional and pooling layers in the lower parts of the network to analyze the local information better than other CNN-based models in steganalysis by adding additional convolutional layers. Second, instead of being positioned before the fully connected layer, the (GAV) pooling layer is placed before the softmax layer, which is the best position for steganalysis. The experiments illustrated that the proposed technique satisfies the best performance compared with some other CNN-based steganalysis methods. The satisfied results are ranged from 57.7% to 81.79% and from 63.91% to 86.17% for S-UNIWARD and WOW steganographic methods, respectively, at 10%, 20%, 30% and 40% bpp embedding rates.

In 2021, Liu et al. [27] proposed the DFSE-Net architecture, which consists of the best-designed architecture of different filters and Squeeze-and-Excitation that may best reflect embedding artifacts. The proposed network has integrated many present suggested designs, such as ABS, BN, and TLU, to create a suitable architecture that outperforms modern methods. According to the authors, DFMs may be used in order to capture more steganographic traces in a variety of ways, and SEMs can be used to increase the relevant features obtained from DFMs. The results

demonstrate that this technique is more effective and performs better. This method requires input images with a specified size. The evaluation results performed using S-UNIWARD, and WOW steganographic methods at 10%, 20%, 30% and 40% bpp embedding rates. The results ranges from 57.8% to 78.35%, and from 65.08% to 85.1% for each method, respectively.

Table (1.1) illustrates the summary of all related works that used BOSSbase (Break Our Steganography System base) [28] as a base to build steganographic datasets, including the techniques used for feature extraction, the number of extracted features, Algorithms used for FS, the number of selected features, and the dataset used.

Table (1.1): Summary of Related Works.

Ref. No.	Feature Extraction Technique	No. of Extracted Features	Algorithm of FS	No. of selected features	Dataset
[14]	CNN with six groups of layers characterized by using: (1) ABS layer; (2) applying the *TanH* activation function at early stages; (3) BN before each nonlinear activation layer.	256	---	---	BOSSbase ver. 1.01
[15]	CNN, which is characterized by using: (1) BN to improve statistical modeling in the subsequent layers; (2) *TanH* to prevent overfitting; (3) Reduce the strength of modeling by using 1×1 convolutions in deeper layers.	128	---	---	BOSSbase ver. 1.01
[17]	second-order co-occurrence matrix applying on TLBP operator where TLBP is applied on residual images generated by many high-order derivative filters.	29,040	---	---	BOSSbase ver. 1.01

[19]	CNN is characterized by process information diversely by using diverse activation modules to learn steganographic artifacts in various ways.	256	---	---	BOSSbase ver. 1.01
[20]	Residual weighted co-occurrence Matrix, where the weights provided by estimation embedding change between image pixels.	34,671	--	--	BOSSbase ver. 1.01
[21]	Extract the features based on analyzing the dependency of image adjacent data utilizing co-occurrence matrix, and using it for image pre-classification as multiple clusters by the K-means Algorithms.	By using K-Means Algorithm with Threshold=20, and K=3 the resulted quantity are 1699, 4405 and 3896 for each cluster, respectively	Select the features that satisfy best performance for each cluster based on a threshold, which based on Euclidean distance analysis.	---	BOSSbase ver. 1.01
[22]	SPAM method	686	Binary Bat Algorithm (BBA)	136 for Hill 132 for WOW 110 for HUGO	BOSSbase ver. 1.01
[24]	CNN architecture uses 30 trainable (self-learning) high-pass filters in the first convolutional layer at the beginning of the architecture.	128	--	--	UCID and BOSSbase version 0.92
[25]	CNN is characterized by improving a pre-processing filter by self-learning and incorporating the knowledge of the selection channel using the embedding probability of each pixel.	256	---	---	BOSSbase ver. 1.01

[26]	CNN, which is characterized by: (1) adding additional convolutional layers in the CNN lower part; (2) new arrangement of convolutional layers and pooling layers; (3) using GAV pooling layer and placed it in a better position.	2 features resulted from applying GAVG on two Feature Maps, which introduces to Softmax for probability calculation	---	---	BOSSbase ver. 1.01
[27]	CNN is characterized by: (1) 30 filter banks for pre-processing. (2) using DFMs to get additional steganographic traces. (3) using SEMs to improve the DFMs features.	240	---	---	BOSSbase ver. 1.01

1.3 Problem Statement

The quantitative analysis of changing cost based on distortion function and the embedding based on Syndrome Trellis Codes (STCs) are two aspects of the content-adaptive steganography. These two aspects make traditional steganalytic techniques were unable to detect content-adaptive steganography effectively. Although approaches based on rich model features are generally good, they are still poor when it comes to identifying images with low embedding rates or high complex texture, or detected with low accuracy.

In addition, most feature extraction methods generate some features that obscure the information provided by the important features that related to be effective in decision-making. These redundant or irrelevant features is due to how the architecture was designed and to some of the hyperparameters that need to be tuned manually.

As a consequence, developing a steganalysis methods, which are updated and independent of any steganographic method and without any previous clue about the hidden content or the embedding method has become essential.

1.4 Dissertation Objectives

The task of steganalysis is to design a blind method that determines whether the image contains a secret message. To achieve this overall task, the following objectives has been established.

1. Design image steganalysis method capable of detecting the presence of secrete information, even with low embedding rates, from various content-adaptive steganographic Algorithms, with notable accuracy and fewer selected informative features subset.

2. Build image datasets for steganalysis purposes, including images embedded at different embedding rates and using different content-adaptive steganographic Algorithms.

3. Design a feature extraction model to extract features that reflect an image's locality and diversity characteristics.

4. Propose a method to preserve the most informative feature subset by eliminating redundant and irrelevant features, and hence, increasing the quality of the feature space and improving detection accuracy.

5. Prove that combining the proposed method for feature extraction and the proposed method to eliminate redundant and irrelevant features, and traditional machine learning as classifiers can be effective in image steganalysis.

1.5 Challenges

Many challenges have been identified when attempt to satisfy the proposed method for steganalysis that can be described as follows:

1. There are no standard image datasets available as a cover and corresponding stego images with different embedding rates used for steganalysis, and this needs extra efforts to build a dataset by applying different content-adaptive steganographic methods on images with different embedding rates.

2. The secrete information of Content-adaptive steganographic is difficult to detect at low embedding rates or detected with low accuracy. This making blind steganalysis is increasingly challenging for steganalyzers.

3. Blind image steganalysis can be seen as a classification problem. The feature space of a classification problem is a crucial factor affecting the performance of a classification/learning Algorithm. Without prior knowledge, it is difficult to identify which features are significant.

1.6 Dissertation Contributions

This dissertation aims to solve some challenges facing practical and universal steganalysis of content-adaptive image steganography. The significant contributions are listed below:

1. Build image datasets for steganalysis purposes, including images embedded at different insertion rates, using HILL, HUGO, MiPOD, S-UNIWARD, and WOW steganographic methods.

2. Develop a feature extraction model based on CNN with two subnets and different kernel sizes to diversify the extracted features that reflect an image's locality and diversity characteristics.

3. Improve the extracted features by developing a FS model based on a two-objective optimization Algorithm that satisfies less selected features and high detection accuracy. This can be satisfied by using Binary Particle Swarm Optimization (BPSO) based on the detection accuracy of Logistic Regression (LR) classifier as a fitness function to preserve with most informative features.

1.7 Dissertation Outline

The remainder of this dissertation is organized as follows:

Chapter 2, presents a basic introduction to steganography, steganalysis, CNN fundamental concepts in addition to the some traditional machine learning classifiers and Particle Swarm Optimization (PSO) algorithm with its binary version.

Chapter 3, provides the detailed methodology of the proposed system. It presents the steps taken, including pre-processing, feature extraction, FS, and classification.

Chapter 4, discuss the implementation of the methods proposed in chapter three for content-adaptive image steganalysis, the experiments result of these methods, and evaluation of the results.

Chapter 5, presents the conclusions of the dissertation and suggestions for future work.

CHAPTER TWO

THEORETICAL BACKGROUND

CHAPTER TWO

THEORETICAL BACKGROUND

2.1 Introduction

This chapter provides a theoretical background required to comprehend the approaches presented in the next chapter. The first sections offer an overview of the basic concepts, families, and Algorithms used in steganography, focusing on content-adaptive image steganography. Then, it describes its counterpart, steganalysis, and explore its many types, down to utilizing CNN and its core ideas. This chapter also discusses the fundamentals and concepts of traditional machine learning classifiers and swarm intelligence for FS, namely PSO and BPSO.

2.2 Image Steganography

Digital media has recently emerged as a vital communication channel carrier as the digital world has expanded, and this was the beginning of a birth of modern steganography [29]. The objective of current steganography is to hide relatively large secret information in a digital medium known as a cover, so that the resulting medium, known as stego, seems similar to the cover to the naked eye. This indicates that an eavesdropper would miss the presence of the hidden message in the stego. The secret message can also be anything that can turn into a bit-stream, such as text, encrypted text, or an image. Cover medium comes in various formats, including image, sound, video, text, and so on. Nonetheless, because images are widely shared over the internet, they are the most commonly utilized medium in steganography and have bits in their digital representation and high degree of redundancy, making them the ideal

carrier for hiding the information [30]. Figure (2.1) illustrates the encoding and decoding procedures of a simple steganographic model of modern stenographic methods.

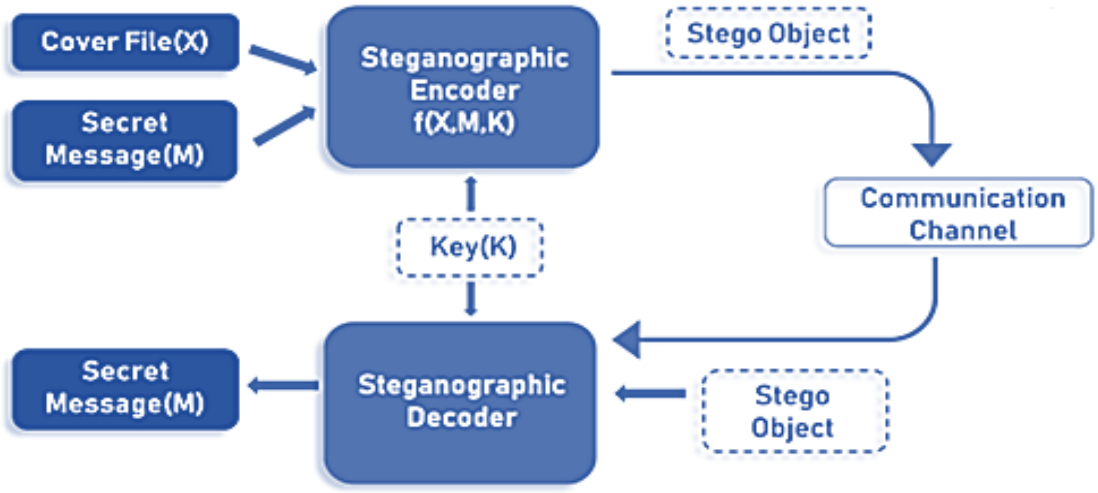


Figure (2.1): The Framework for Encoding and Decoding a Steganographic Model[31].

• **Encoding Process:** As input, both the cover file (X) and the secret message (M) are given into the steganographic encoder. The steganographic encoder function, f(X, M, K), embeds a cover file with the hidden message. The resulting stego object is identical to your cover file, with no discernible differences.

• **Decoding Process:** In this stage, stego object is passed into steganographic decoder to extract the hidden message.

However, given the presence of an eavesdropper, these tasks can be challenging. The eavesdropper should be unaware of any communication between the sender and the receiver, and otherwise, he will break the communication channel.

This dissertation is primarily concerned with spatial content-adaptive steganographic methods that employ grayscale images as the transmission medium.

2.2.1 Image steganography Families

There are three prominent families of steganography. Based on how the secret information is hidden, several families were identified, and they are as follows [32]:

- Cover Selection Steganography.
- Cover Synthesis Steganography.
- Cover Modification Steganography.

These three families of steganography will be discussed in the following sections, highlighting their benefits and disadvantages.

2.2.1.1 Cover Selection Steganography

In this steganography, the sender has a predetermined image database, and the sender chooses the one that best fits the intended message from this database. In this scenario, the sender uses message digesting functions to search the image database for an image with a digest that matches the bit-stream requested in the message. When the image is identified, it is sent to the receiver, who can easily interpret the hidden message by repeating the hashing Algorithm with the specified secret key.

Such methods have the benefit of being almost undetectable [32]. Indeed, because the cover is not modified in any way, it is difficult to identify that there is a secret message. However, the primary issue with

these approaches is their extremely low embedding capacity, which immediately becomes impractical since the number of tries necessary to discover a match rises exponentially with the size of the digest [33].

2.2.1.2 Cover Synthesis Steganography

In this steganography, the sender creates a cover that best conveys the hidden message. If the sender can create a cover with a known distribution with the receiver, it will keep its secret message hidden safely.

This method has the advantage of providing a high degree of security. Nevertheless, because of its restricted embedding capabilities, it is a less desirable solution [32].

2.2.1.3 Cover Modification Steganography

The most frequently used and researched steganography approach is cover modification steganography. Its basic idea entails modifying an existing cover object to hide the information using a shared secret key and, at the same time, attempting to maintain, as much as possible, the natural statistics of the utilized cover. Figure (2.2) depicts the fundamental idea of cover modification steganography, where *Emb* and *Ext* represent the embedding and extraction map functions.

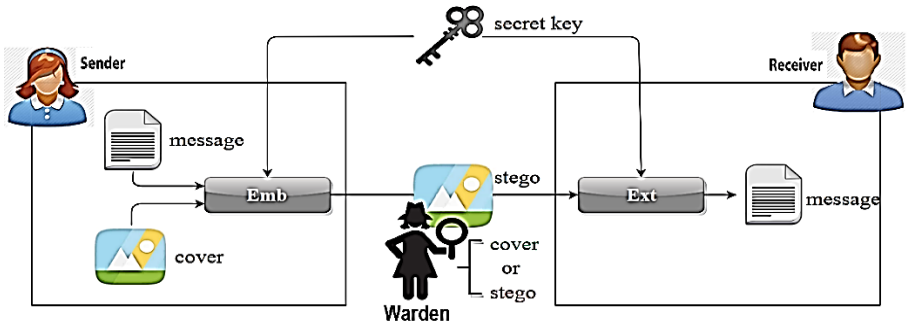


Figure (2.2): General Diagram of Cover Modification Steganography [34].

2.2.2 Adaptive Steganography

Since the Syndrome Trellis Codes (STCs) approach [35], it has been possible to code a message while considering the image's content; the first method that employs this coding in 2010 is HUGO adaptive steganography method [36] that considered a most secure manner. It involves hiding the information while attempting to induce the cover image with the least degree of distortions. Adaptive steganographic Algorithms try to hide the information in this regions of cover image that difficult-to-detect. Because cover modification steganography disturbs the original statistics of the cover through the hiding process, it is essential to choose secure regions of the cover that have no noticeable influence on the total cover statistics. Most adaptive techniques utilize a distortion function, which simulates the embedding influence on detectability. Compared to model-preserving approaches [37], adaptive steganography methods provide better generalization and even a higher level of security. Furthermore, it permits the creation of steganographic Algorithms that are motivated by the performance of a steganalyzer in terms of security.

The following three stages are taken into account by adaptive steganography Algorithms:

1. Determine the optimal position for embedding.
2. Message encoding.
3. Embedding the message.

2.2.2.1 Determine the Optimal Position

Adaptive steganographic Algorithms try to hide the information while minimizing the effect of the embedding process [38]. To attain this

goal, it is necessary to develop a distortion indicator capable of evaluating the detection capability induced by the hiding process.

The overall embedding effects generated by the hiding process are represented as a distortion by a steganographer. Adaptive steganographic methods hide the information based on function minimizing to preserve as much of the cover medium as possible. In the case of grayscale images, it is represented by the following mathematical function [39]:

$$D: \{0, \dots, 255\}^n \times \{0, \dots, 255\}^n \to [0, \infty) \,, \tag{2.1}$$

where D represents the total distortion, it should approximate the detection capability produced by the hiding process. The function that estimates the difference between the cover and stego images in feature space or another space can be used to characterize the hiding impact [40]. Let $x = (x_1, \dots, x_n) \in$ cover with $x_i \in \{0, \dots, 255\}$ a cover image comprised of n components and y = (y_1, ..., y_n) $\in$ Stego with $y_i \in \{0, \dots, 255\}$ the corresponding stego image. m represents the secret message comprised of m elements, with $m = (m_1, \dots, m_m) \in \{0, 1\}^m \in M$. The distortion function D can then be expressed as follows [39]:

$$D(x, y) = \|f(x) - f(y)\|, \tag{2.2}$$

where f(x) and f(y) represent the features vector, that characterizes the cover and stego images.

The distortion function in Equation (2.2) provides a main challenge for the steganographer since it is too complicated to find a code that minimizes it. Two techniques have been presented in an attempt to overcome and simplify this challenge.

The first proposed approach assumes that modifying a pixel does not affect detectability of neighboring pixels, allowing the distortion to be approximated in an additive version. This viewpoint is shared by the authors of [35, 41], among many others. They offer an additive distortion version that is related to the utilization of a cost map ρ. It consists in assigning a value $\rho_i \in [0, \infty)$ to each pixel of the cover x_i. This value indicates the impact of modifying the i^{th} pixel on global security as described in Equation (2.3) [11]. The case where $\rho_i = \infty$ suggests that the pixel x_i cannot allow to being changed.

$$\rho_i = D(X, X{\sim}X_i) \tag{2.3}$$

In Equation (2.3), $X{\sim}X_i$ represents the cover image whose i^{th} pixel has been changed.

The purpose of adaptive steganography is to reduce the influence of the hiding process that is measured using a predefined distortion function. As it stands, it is easy to infer that the most challenging aspect of this approach is calculating the cost map ρ.

The second suggestion is to divide the global problem into smaller sub-problems that may be easy to handle. The authors of [35] suggested a novel distortion function. Their method works by arranging the cover pixels into sub-lattices of pixels, each of which is independent of the others. Those sub-lattices are denoted by $S = \{S_1 \cup ... \cup S_n\}$. The message is then divided into little parts, hiding in a sub-lattice to minimize a distortion function. The employed distortion function is a total of local distortions computed on cliques.

2.2.2.2 Message Encoding

STCs [35] are a common coding method for embedding hidden messages into cover images while achieving the best coding performance. Under the steganography paradigm of distortion minimization, it may be utilized to address binary and non-binary embedding problems. The message embedding can be expressed as follows for the binary problem[39]:

$$Emb(X, m) = \arg \min_{P(Y) \in C(m)} D(X, Y) \tag{2.4}$$

The embedding function is denoted by Emb(.), and the secret message is denoted by m. The cover and stego, respectively, are $X = (x_1, x_2, x_3, \ldots, x_n)$ and $Y = (y_1, y_2, y_3, \ldots, y_n)$. $C(m)$ is the stego's feasible set, which must satisfy the following formula [39]:

$$C(m) = \{y \in \{0,1\}^n | \mathrm{H}y = m\} \tag{2.5}$$

The parity-check matrix, which is considered the key to the entire hiding and extraction process, is denoted by H. Set the secret message length to m. Parity-check matrix is formulated by m placing the sub-matrices $\hat{\mathrm{H}} \in \{0, 1\}^{h \times w}$ next to each other and shifts them down one row, resulting in a sparse, banded matrix H. The easiest way to explain the encoding process is to use a parity-check matrix syndrome trellis. The blind message extraction process is formulated as follows [39]:

$$\mathrm{Ext(m)} = \mathrm{H}\, P(Y)\ , \tag{2.6}$$

where Ext(.) represents the message extraction function derived by multiplying the parity-check matrix by the stego's parity function P. Without malicious attacks, the STCs framework can effectively retrieve

embedded messages. For additional information about STCs, the references [41, 42] provide more information. The example in Figure (2.3) is a simplified illustration of the workings principle of the trellis approach (STC).

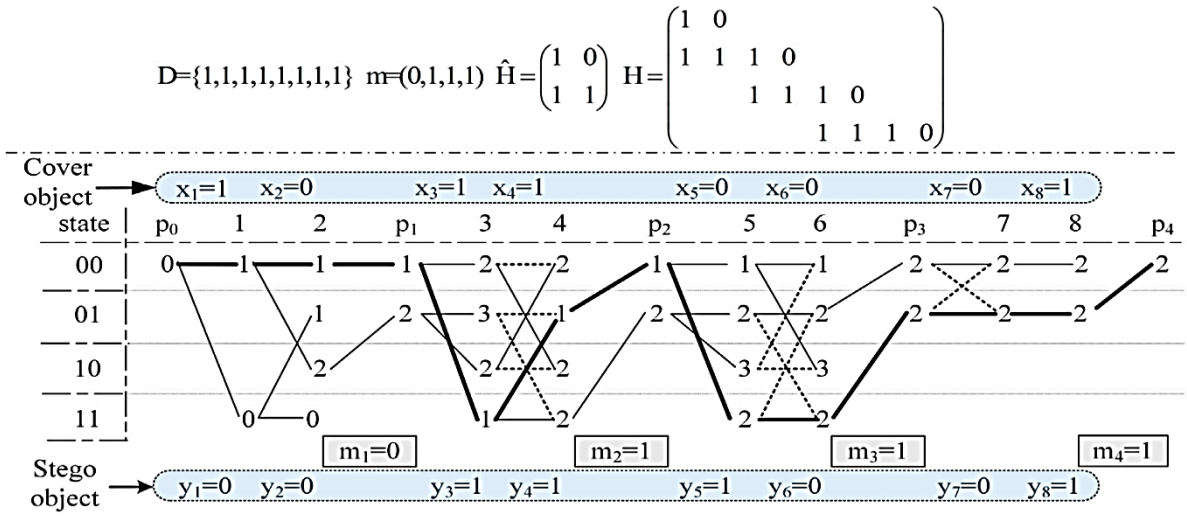


Figure (2.3): STC Process [43].

2.2.2.3 Embedding the Message

The embedding procedure is accomplished by modifying the cover image *X's* selected pixels (after syndrome coding has been applied). By performing LSBs replacement or matching to some selected pixels, the stego vector is embedded into the cover image.

1. Steganography By LSB Replacement: This technique is the most widely utilized embedding approach for practical reasons. As illustrated in Figure (2.4), this technique consists in replacing the least significant bits (LSBs) of the selected pixels with the bits of a message to be embedded (stego vector). In other word Using this method, the LSBs of the selected pixels are replaced with the bits of a secret message. It is

significant to mention that can be modified by changing more than one bit of the LSB; this is known as "2LSB replacement."

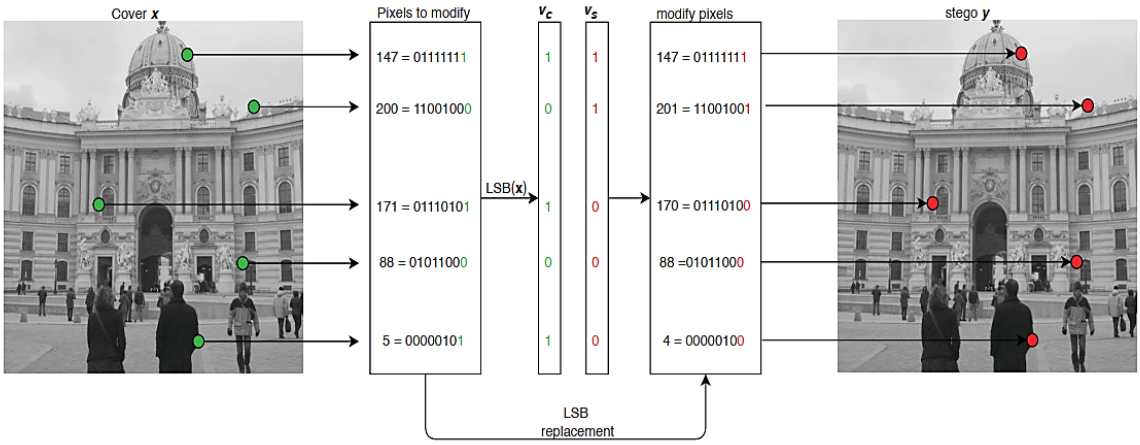


Figure (2.4): Embedding Process using LSB replacement [39].

However, LSB replacement steganography is not very secure [36]. Indeed, even if the modifications performed have little effect on the medium's visual appearance, they significantly impact its statistical distribution. As a result, a simple statistical analysis would be adequate to identify the modification [39]. Machine learning [44, 45] and deep learning [46, 47] are highly efficient for detecting very low payloads.

2. Steganography By LSB Matching: This technique hides the stego vector *Vs* into the cover pixels and quite similar to the LSB replacement. Instead of changing the LSB pixels, this method operates by randomly increases or decreases the pixel's value by 1. Figure (2.5) illustrate the embedding algorithm using the LSB matching, also known as ±1 embedding. The author in [47] introduces the first LSB matching hiding technique, which proposed as a solution to the LSB replacement's low security. LSB matching technique methods are more challenging to be detected than LSB replacement methods. It is important to point out that all modern and efficient Algorithms rely on "LSB matching" hiding.

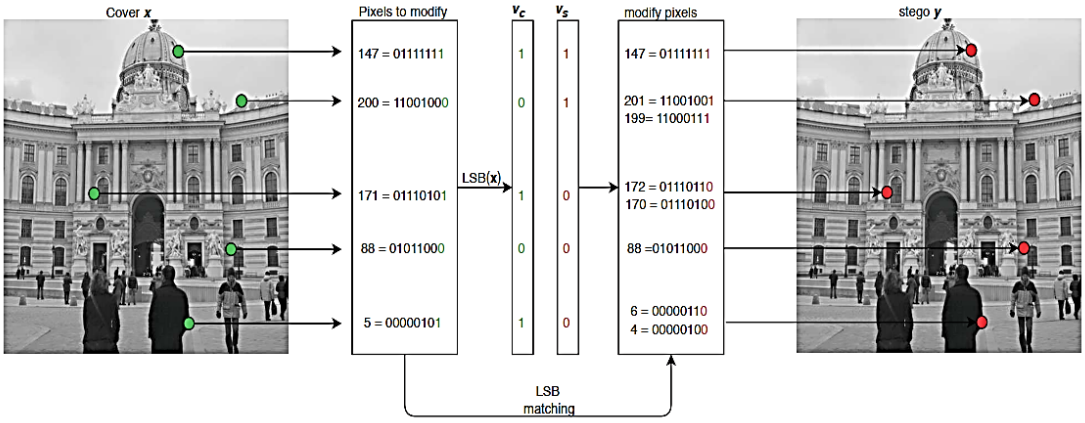


Figure (2.5): Embedding Process using LSB Matching [39].

Figure (2.6) depicts an adaptive steganography algorithm where x represents the grayscale cover image. The cover vector is represented by v_c, which is defined as $LSB(x) = v_c = (v_{c1}, ..., v_{cn}) \in \{0, 1\}^n$. v_c' and ρ' are the results of shuffling v_c and ρ using the shared secret key, respectively. v_s represents a stego vector such as $v_s = (v_{s1}, ..., v_{sn}) \in \{0, 1\}^n$ that produced by applying STC. A replacement or matching embedded is required to obtain the stego image y that resulted from embedding v_s within the cover image x.

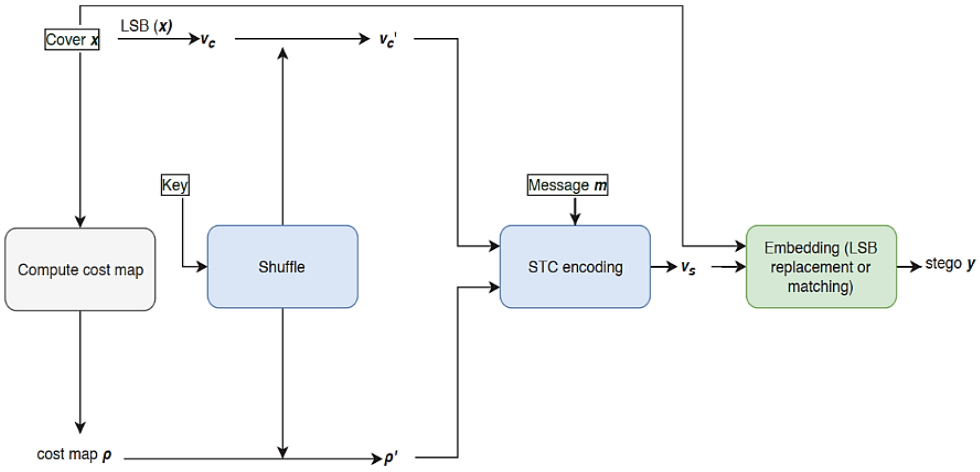


Figure (2.6): Embedding Framework in Adaptive Steganography[39].

2.2.3 Embedding Algorithms in Spatial Domain

Embedding methods in the spatial domain perform the embedding process directly on the pixels. As a result, the embedding rate (the quantity of data embedded for a particular cover) is expressed as a bit per pixel (bpp). When compared to transform domain approaches, these techniques provide more embedding capacity and require less processing time.

This section describes the steganographic algorithms employed in this dissertation for validation, including five content-adaptive steganographic methods.

2.2.3.1 HUGO Algorithm

HUGO algorithm [36] is considered one of the first adaptive algorithm that has been found in the literature. It was proposed in 2010 and utilized in the BOSS competition the same year as a hiding method [28]. HUGO's cost function computes the weighted sum of differences between feature vectors taken from a cover image and its stego counterpart in SPAM [24] feature space.

The cost map $\rho = \{\rho i \in [0, \infty)\}_{i=0}^{n}$ of the HUGO algorithm is defined such that, for the *i*th pixel, the cost of its modification is [36]:

$$p_i = \sum_{j=1}^{d} w[j] \left| f_x[j] - f_{x \sim x_i}[j] \right| \qquad (2.7)$$

with:

- *f* is a feature vector generated from the co-occurrence matrix. Each *bin* from this matrix contains the number of occurrences of values triplets ($d1$, d_2, d_3) in the residual image. This latter is obtained by filtering the

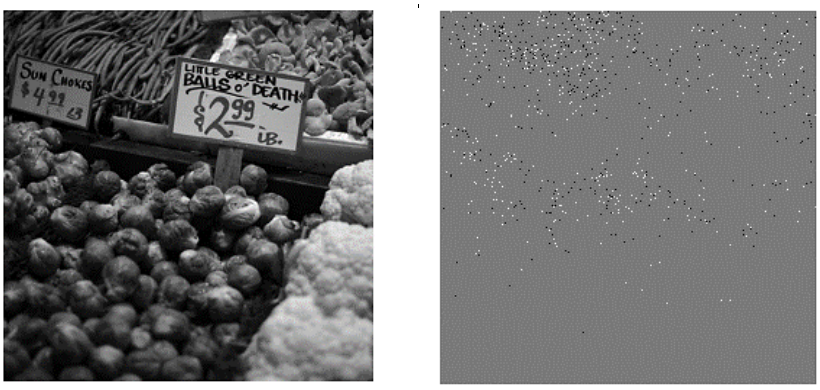
image with the kernel [1- 1], followed by a truncation in the interval $\{-T, ..., T\}$.

- f_x refers to d-dimension feature vector of the image x.
- $f_{x\sim x_i}$ refers to the features vector of the image x whose ith pixel has been modified.
- $w[j]$ is the weight associated with the triplet $(d_1, d_2, d_3) \in \{-T, ..., T\}$, it is calculated as follow[36]:

$$w[j] = \frac{1}{\left[\sqrt{d_1^2+d_2^2+d_3^2}+\sigma\right]^{\gamma}}, \tag{2.8}$$

with σ and γ are two parameters that can be tuned to minimize the detectability. The term $w[j]$ is used to promote the embedding in images zones where the triplet $(d_1, d_2, d_3) \in \{-T, ...T\}$ is high, which tends to favor embedding in textured or contour zones.

Once the cost map ρ is calculated, go to the following stage, where the pixels to be modified are chosen using either a simulator or STCs. After that, these pixels are modified by ± 1 based on the amount of



(a) (b)

Figure (2.7): Pixel Modification a) Cover image b) The resulting of embedding changes after applying the HUGO Algorithm: +1=white -1=black.

distortion induced by each. Figure (2.7) illustrate an example of pixel modified for HUGO algorithm by using 10% embedding rate applied to the same image along with the resulted distortion.

2.2.3.2 WOW Algorithm

The WOW embedding algorithm works as follows [48, 49]: Firstly, three directional filters (denoted $K^{(k)}$, $k = 1, 2, 3$ using Daubechies 8 wavelets) are performed on the cover image X to obtain the *LH, HL* and *HH* directional residuals $R^{(k)} = K^{(k)} * X$, respectively. The convolution mirror-padded operation is denoted by $*$ symbol. And then the embedding suitability $\xi_{ij}^{(k)}$ for each pixel can be obtained by measuring the difference between $R^{(k)}$ and the same residual after changing only one pixel at *ij* (denoted $\xi_{[ij]}^{R(k)}$) by the wavelet coefficient itself [48].

$$\xi_{ij}^{(k)} = |R(k)| * |R(k) - R_{[ij]}^{(k)}| \tag{2.9}$$

Then the embedding costs ρ_{ij} are computed by aggregating three suitability $\xi_{ij}^{(k)}$, $k = 1, 2, 3$ by [48]:

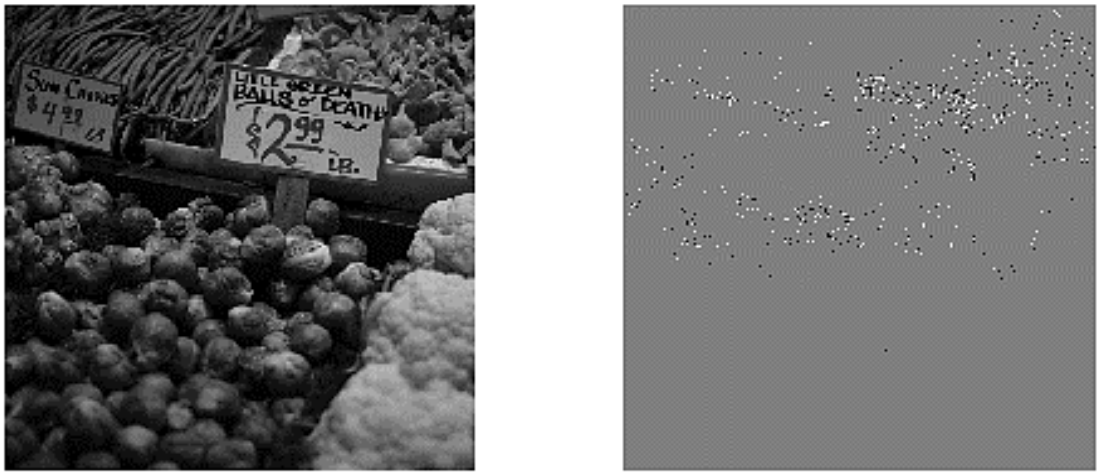
$$\rho_{ij}^{(p)} = \left(\sum_{i=1}^{3} \left|\xi_{ij}^{(k)}\right|^{p})^{-1/p}\right), p = -1 \tag{2.10}$$

Finally, the STCs is applied to minimize the following distortion function and get the resulting stego image Y. [49]

$$D(X,Y) = \sum_{i}^{n_1} \sum_{j}^{n_2} p_{i,j}(X, Y_{i,j}) \left|X_{i,j} - Y_{i,j}\right| , \tag{2.11}$$

where $n_1 \times n_2$ denotes the dimension of cover image X; $\rho_{i,j}$ denote the costs of changing pixel X_{ij} to Y_{ij} , the WOW limits the embedding changes to ±1.

Usually, the $\rho_{i,j}^{(-1)}$ is smaller for those pixels located at the regions with more textural complexity, and thus the modifications after minimizing the above distortion function with STCs would be concentrated on textural regions, and the use of STCs can significantly reduce the embedding changes compared with typical methods, such as LSB Matching, the WOW is currently the most secure steganography in spatial domain. Figure (2.8) illustrate an example of pixel modified for WOW algorithm by using 10% embedding rate applied to the same image along with the resulted distortion.



(a) (b)

Figure (2.8): Pixel Modification a) Cover image b) The resulting of embedding changes after applying the WOW Algorithm: +1=white -1=black.

2.2.3.3 S-UNIWARD Algorithm

S-UNIWARD [11] is a spatial embedding Algorithm that is comparable to the WOW technique. The UNIWARD distortion function is used in this technique to embed a secret message in the spatial domain.

Pixel costs are determined from three directions, like in the WOW method.

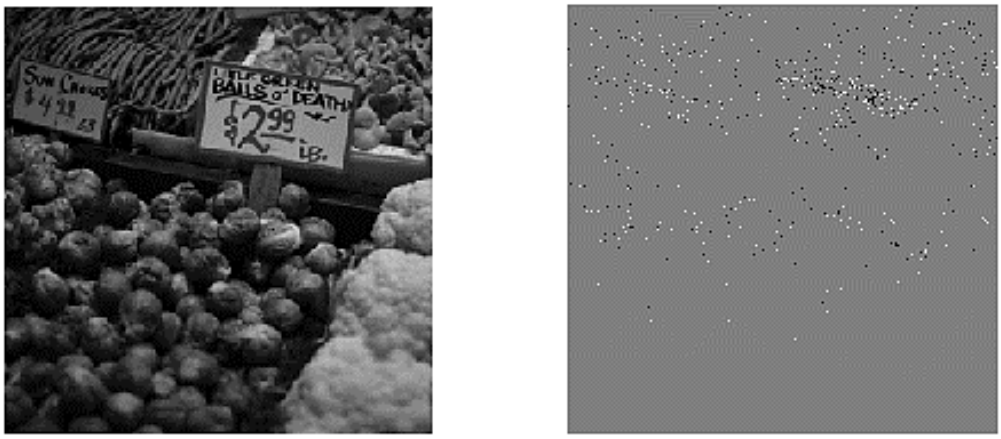
In S-UNIWARD the cost map ρ is composed of $\{\rho_i \in [0,\infty)\}_{i=1}^{n}$, where ρ_i is given by the following equation [11]:

$$\rho i = \sum_{k=n}^{d} \sum_{u=1}^{n_1} \sum_{=1}^{n_2} \frac{\left|W_{uv}^{(k)}(x) - W_{uv}^{(k)}(x \sim x_i)\right|}{\sigma + \left|W_{uv}^{(k)}(x)\right|}, \quad (2.12)$$

with

- $W_{uv}^{(k)}$ the wavelet coefficient in the $(u, v) \in \{1, .., n_1\} \times \{1, ..., n_2\}$ position for the *kth* sub-band of the image x,
- $W_{uv}^{(k)}(x \sim x_i)$ the wavelet coefficient in the $(u, v) \in \{1, .., n_1\} \times \{1, ..., n_2\}$ position for the *k*th sub-band of the image x whose pixel *i* has been modified,
- σ a numerical stabilization constant.

Note that there are three filtering directions for wavelet decomposition: horizontal, vertical, diagonal. A closer look at the equation shows that the cost ρ_i is small in textured areas. Indeed, for a pixel *i*, the intensity variations in the vertical, horizontal, and diagonal directions are obtained via wavelet decomposition. The higher the amplitude is in one or more directions, the larger the denominator is, and therefore the smaller the ρ_i; this indicates that it is possible preferably choose this pixel to make a modification. Figure (2.9) illustrate an example of pixel modified for S-UNIWARD algorithm by using 10% embedding rate applied to the same image along with the resulted distortion.



(a) (b)

Figure (2.9): Pixel Modification a) Cover image b) The resulting of embedding changes after applying the S-UNIWARD Algorithm: +1=white -1=black.

2.2.3.4 HILL Algorithm

HILL [6] is a spatial embedding technique that was developed in 2014 as an improved version of WOW. The authors when analyzing the equation of calculating the cost map, have discovered the presence of some pixels is given a high cost within texture areas; this is due to the pixels' predictability in one of the three directions. Therefore, the pixel in a texture region, on the other hand, should be allocated a lower cost value even if it is predictable in one of the directions. To achieve this aim and ensure that the pixels inside the textured areas always have the lowest possible cost, they replaced the horizontal, vertical, and diagonal directed kernels with a single non-directed high-pass filter, followed by two low-pass filters.

Based on the above analysis, improved cost map have proposed that given by the following equation [6]:

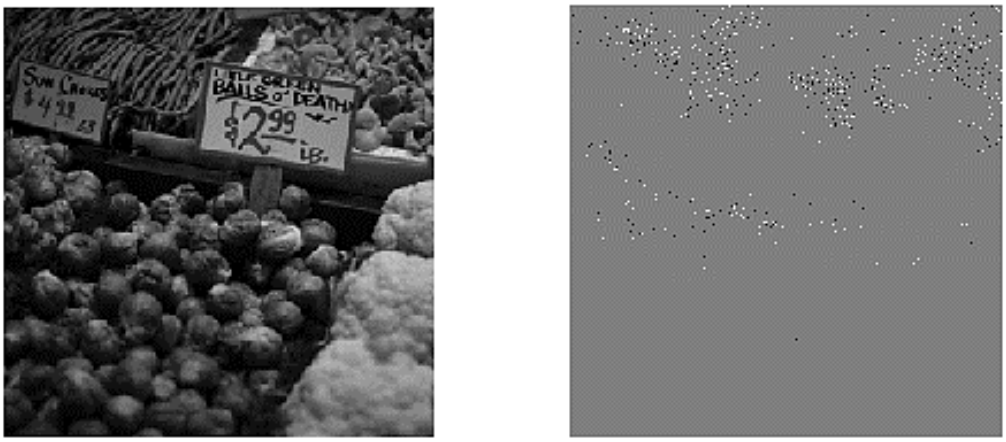
$$W^{(k)} = \mathrm{x} \star H^{(k)},$$

$$\xi^{(k)} = |W^{(k)}| \star L_1, \tag{2.13}$$

$$\rho = \sum_{k=1}^{d} \frac{1}{\xi(k)} * L_2$$

with $W^{(k)}$ is the residual obtained by convolution the image x with a high pass filter $H^{(k)}$. $\xi^{(k)}$ is the embedding suitability.

The high-pass filtering is used to discover the less predictable areas of an image, while the two low-pass filtering are used to make the low-cost values more clustered. Figure (2.10) illustrates an example of pixel modified for HILL algorithm by using 10% embedding rate applied to the same image along with the resulted distortion.



(a) (b)

Figure (2.10): Pixel Modification a) Cover image b) The resulting of embedding changes after applying HILL Algorithm: +1=white -1=black.

2.2.3.5 MiPOD Algorithm

In 2016, the MiPOD Algorithm [50] was proposed. It is one of the most secure spatial domain embedding methods. This method is

significantly different from the preceding techniques in that there are no pixel costs begin with it. Instead, based on a residual model, the embedding change rates p ={p_1, ..., p_n} are first computed by solving numerically the two following equation [50]:

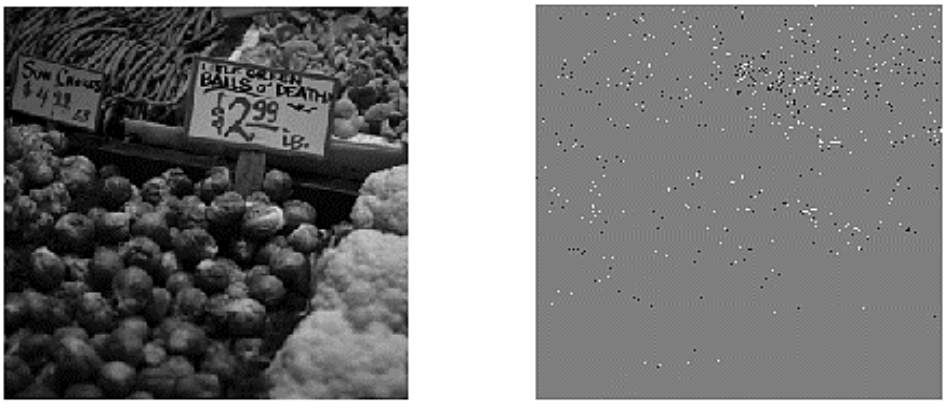
$$p_i\sigma_i^{-4} = \frac{1}{2\lambda}\ln\frac{1-2p_i}{p_i}, i = 1, \dots, N, \quad (2.14)$$

with σ_i^2 is the variance of the cover's pixel x_i, that is computed using a variance estimator. λ is the Lagrange multiplier.

Once the change rates are computed p, using the method of Lagrange multipliers, they are converted to costs ρ = {ρ_1, ..., ρ_n} using the following equation[50]:

$$\rho_i = \ln (1/p_i - 2) \quad (2.15)$$

These costs are then used by the syndrome-trellis codes to embed the payload *R* during the embedding process. Figure (2.11) illustrate an example of pixel modified for MiPOD algorithm by using 10% embedding rate applied to the same image along with the resulted distortion.



(a) (b)

Figure (2.11): Pixel Modification a) Cover image b) The resulting of embedding changes after applying the MiPOD Algorithm: +1=white -1=black.

2.3 Image Steganalysis

Steganalysis is thought to be the opposite of steganography. It attempts to overcome the secrecy of steganography by developing hypothesis tests that may detect minor modifications between the cover and stego images and categorize them accordingly. The aim of the steganalyst/warden is to detect the presence of secret information. There are three types of attacks (malicious, active, and passive attacks), which are as follows [46]: The transmission can be attacked with the intent by generating a fake message (a malicious attack) or by damaging the message (an active attack) or, alternatively, by checks the communication between the sender and receiver to see if it contains a hidden message (a passive attack). In this regard, steganalysis gives the methods necessary to discover hidden information.

The steganalysis system's primary objective is to detect the existence of hidden information in images. In general, depending on prior knowledge about the steganographic method, there are two approaches to steganalysis: specific and blind [51]:

2.3.1 Specific Steganalysis

Also known as targeted steganalysis. This steganalysis system class is based on the technique of attacking a particular embedding method by identifying its security vulnerabilities (statistical flaws). They are sometimes also restricted to specified image format. These systems provide highly precise results, but when tested with other stego images generated by an Algorithm different from the one being attacked, they can ultimately fail.

The attack proposed by [52] is an example of targeted steganalysis. This focused steganalysis technique is designed to identify any HUGO Algorithm [36]. A 4-D feature vector was enough to detect the HUGO method in this approach. Because of the emergence of adaptive steganographic Algorithms, it is nearly difficult to identify an embedding technique with exploitable defects; therefore, this specific steganography is no longer an updated paradigm.

2.3.2 Blind Steganalysis

This is the current favored paradigm for attacking a steganography system. It is also known as Generic or universal steganalysis. Unlike targeted steganalysis, these approaches are not intended to attack a single embedding technique but rather to identify many forms of steganographic methods. Because this kind detects a wide range of steganography techniques, it typically has less accuracy; nonetheless, blind steganalysis is an essential tool for detection if the embedding mechanism is unidentified or secrete.

Blind steganalysis approaches are classified into two types [10]: (1) machine learning-based steganalysis and (2) deep learning-based steganalysis. Steganalyzers in machine learning-based steganalysis utilizes substantial feature engineering to take advantage of the fact that steganographic embedders generate statistically significant artifacts [53]. Deep learning-based steganalyzers, on the other hand, employ CNN architectures to learn the statistical imprints of various steganographic embedders [44, 54]. These approaches are highly successful against spatial and frequency steganography [10].

1. Machine learning-based Steganalysis: This category of steganalysis is regarded as a hypothesis-testing issue in which a specific digital media must identify whether it is "stego" or not. As a result, steganalysis may be seen as a problem with two classes. This means that the combination of steganalysis with machine learning techniques and specific feature engineering to extract rich signal data from the underlying source distribution [53]. In general, statistical steganalyzers are only effective against a limited number of spatial and frequency-based steganographic methods [55].

The feature extraction stage involves transforming the database using the hand-crafted transformation function. The resulting database trains the blind classifier to learn a mapping function that translates feature vectors to stego or cover labels.

Although the usefulness of the learnable classifiers and their direct influence on the success of machine learning techniques, the feature extraction stage is the most essential and challenging stage. The well-designing and effective feature extraction help the steganalyst reduce the search area of the classifier and successfully identify the required classes.

Several studies have been proposed in this regard, including SPAM, SRM [18], PSRM [56], and CCJRM [57], all of which follow the same typical procedure.

The feature vector produced by the feature extraction technique is utilized for training a classifier for the classification phase. SVM [58] and Ensemble classifiers [9] are two of the most used classifiers.

2. Deep Learning-based Steganalysis: The retrieved features in computer vision applications such as object recognition should be flexible

to the object's various orientations, scales, viewing angles, and occlusions. However, it is pretty challenging to construct a hand-crafted features vector capable of handling such complexity; as a result, the efficiency of this framework will be far from optimum.

ALexNet [59] was presented as a novel CNN-based approach for mathematical learning of the features rather than hand-crafted extraction. Since then, the focus of computer vision research has turned away from improving feature extraction Algorithms toward designing self-learning feature techniques.

Deep learning, especially CNNs, is being introduced into the discipline of steganalysis with the aim to learn more relevant and effective features and, therefore, improve performance.

2.4 Convolutional Neural Network

A convolutional neural network (CNN) is a deep learning architecture that can take an input image, assign importance (learnable weights and biases) to various aspects/objects in the image, and then

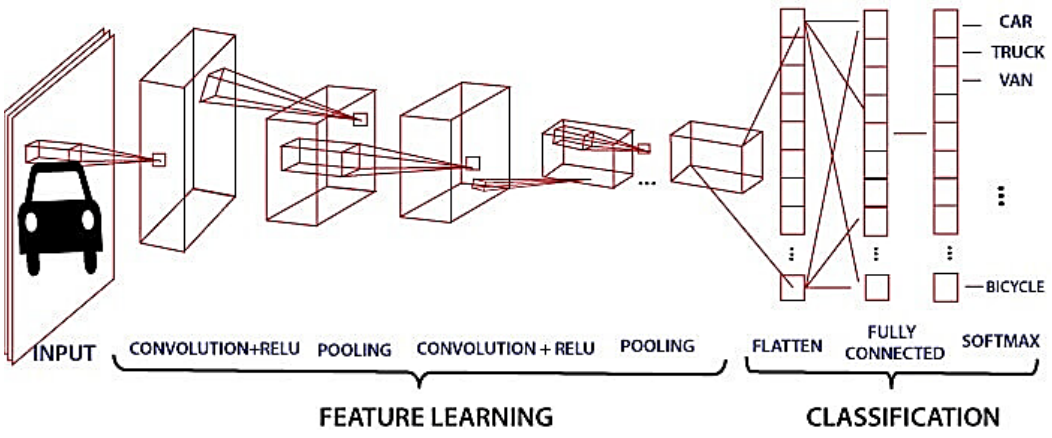


Figure (2.12): Schematic Illustration of a typical CNN Architecture [60].

classify the image using these signals. Figure (2.12) depicts a CNN consisting of multiple specialized layers. The power of CNNs to learn important features necessary to identifying steganographic images has made them popular in steganalysis [10].

A CNN architecture, as shown in Figure (2.12), can be divided into two parts: feature learning or convolutional module and classification module. The input layer receives the input image and passes it to the feature learning module. This module consists of several blocks, each of which has some layers. It works as a feature extractor by turning an image into a feature vector. The classification module's input is then linked to this vector. The classification module is made up of many layers that are fully connected. The classification module's goal is to classify the input image by combining the features that have been gathered. The CNN output will be provided by this module's last layer (the prediction). The resulting numerical values are typically normalized between [0, 1], using the softmax function. A more detailed review of the field can be found in [61].

More details on the layers and some concepts related to the CNN are described in the following sub-sections.

2.4.1 Input Layer

The input layer [62] includes image data, where the raw image input data is received and stored for processing in the network. The width, height, and the number of channels are specified by this input data. For RGB values, the number of channels is typically three for each pixel, while for grayscale values, the number of channels is one for each pixel. The input layer with 3D volume is depicted in Figure (2.13).

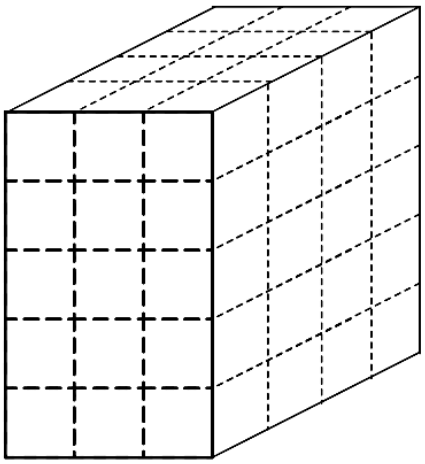


Figure (2.13): Input Layer 3-D volume.

2.4.2 Convolution Layer

This layer is considered the main component of CNNs [61]. Its purpose is to use a convolution operator to detect a set of features in the input image. Suppose I represents an image passed into the CNN's input layer, and K represents a filter with a size of $(n \times m)$ that is used to convolves the image. The objective of convolution is to replace each pixel with a weighted sum of its neighbors, which defines what is known as a kernel. This operation produces an output I_f based on the following Equation:

$$I_f = I * K , \tag{2.16}$$

where * represents a 2D convolution.

A FM is a term used to describe the output of a convolution layer [63]. Parameters like stride and padding determine how much information is included in a FM following each convolution. Padding surrounds the image size with zeros, whereas stride represents the step size the convolution filter takes across the input image. If stride=1, the filter K will slide pixel by pixel. The filter slides across the input at increasing intervals

when the stride size increases, resulting in the reduced overlap between the pixels. The result of the convolutions is added to the FM as the filter moves over the input image. Figure (2.14) depicts the example of sliding process across the image pixels.

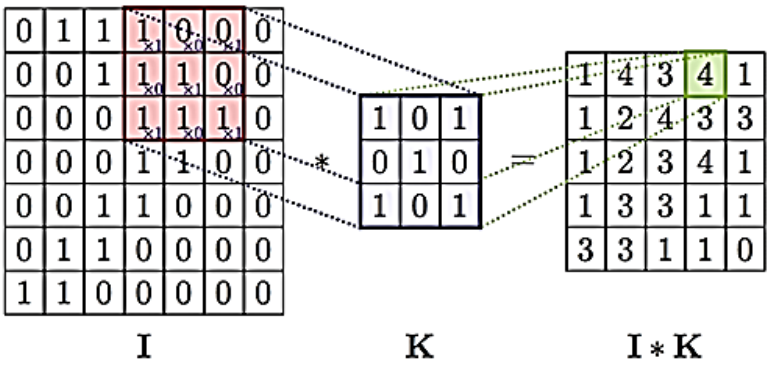


Figure (2.14): Example of Convolution Operation.

2.4.3 Activation Layer

The CNN should be able to extract discriminating features for steganalysis by defining the activation function. Every convolution layer on each block is followed by a nonlinear function, and it introduces nonlinearity into the network to take the benefits of using multiple stacking convolution layers. To put it another way, if the CNN without nonlinearity, it would operate like a perceptron with a single-layer regardless of how many the convolution layers it had because stacking linear functions will generate additional linearity. This layer accepts the feature-maps created by the convolutional layer $F^{(l)}$ as input and generates another feature-map termed activation-map $A^{(l)}$. The activation function is considered as an element-wise procedure. As a result, the input features-map and the resultant activation-map have the same dimensions as shown in Equation (2.17) [61].

$$A^{(l)} = f(F^{(l)}), \tag{2.17}$$

where *l* represent the current layer.

Many activation functions, such as *TanH* and Leaky Rectified Linear Unit (*LReLU*), are related to the proposed work, as illustrated in Figure (2.15).

- Tangent Hyperbolique *TanH* [16]: This function's outputs are zero-centered inside the range [-1; 1], making it easy to represents the inputs with positive, negative, and neutral values:

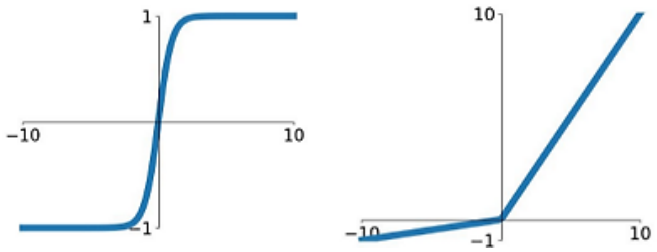
$$TanH(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.18}$$

TanH, on the other hand, has a problem with vanishing gradients.

- Leaky ReLU (LReLU) [65]: The Rectified Linear Unit (ReLU) is the most widely utilized activation function in a CNN. All negative inputs are set to zero by this activation function, as illustrated in Equation (2.19) [66].

$$ReLU(z) = \begin{cases} 0 \; if & x < 0 \\ x \; if & else \; \geq 0 \end{cases} \tag{2.19}$$

The main weakness of this function is what is called the problem of “dying", which happens when a ReLU neuron output for each given input is less than zero, the activation will have a zero final output. When a neuron goes negative, it cannot recover since the ReLU activation will have a zero slope in negative regions. As a result, the particular neuron is unable to interpret the input and is rendered ineffective. After a significant number of epochs of training, a large part of the network may become completely worthless in this manner.



(a) **(b)**

Figure (2.15): Non-linear Activation Functions [64]: (a) Hypebolic Tangent. (b) Leaky ReLU

To overcome dying issues, a new version has been introduced, known as the LReLU activation function, that performs the same ReLU but with a threshold operation by multiplying the output with a scalar $\propto$ (Equation 2.20) when the input value is negative.

$$f(x) = \begin{cases} x, & x \geq 0 \\ \propto * x, & x < 0 \end{cases} \tag{2.20}$$

Experiments show that LReLU has a slight effect on accuracy compared with ReLU. Leaky rectifiers have non-zero gradients across the entire domain and perform similarly to standard rectifier deep neural networks [37].

- Softmax: The softmax function [65] or exponential function often implements at the output layer of neural networks used for classification. It estimates the posterior probability of each class label over *j* classes as a real value within the range [0*;* 1]. This function is a normalized exponential that is defined as follows [65]:

$$Z_i = \frac{e^{x_i}}{\sum_{j=1}^{2} e^{x_j}} \ , \tag{2.21}$$

for $i = 1, 2$, where x_i is the total input to the neuron i in the top layer, and Z_i represents the output.

2.4.4 Pooling Layer

This layer, also identified as a down-sampling layer, reduces the activation-maps' spatial volume. It helps to:

1. Reduce the amount of parameters and the number of calculations required.
2. To make the network more efficient by compacting the activation-maps.
3. Reducing the chance of overfitting [59, 65].

There are no weights to be updated in this class of layers. It is frequently used after a sequence of other layers (i.e., convolution, activation, and Batch Normalization layers). The pooling layer works through sliding a window over each map and lowering the data inside those windows to a single value either by using the max or average. This procedure is continual while sliding ends, and each map has been spatially reduced.

Max and average pooling, as seen in Figure (2.16), are the two most used pooling strategies. Max pooling [68] selects the maximum value within the sliding window and ignores the others. The main goal of this layer is to make the network focus on feature extraction rather than feature location through maintain translational invariance. As a result, image recognition often employs this method of pooling.

In contrast, average pooling (AVG-Pooling) [69] allows to examine all of the values in the FM through taking the average of the values inside the specified window area. This type of pooling enforces the generalization by effectively combining many values one value, reducing

over-fitting risk. However, average pooling is the most widely used method in steganalysis since the embedding information is expected to be hidden in image noise.

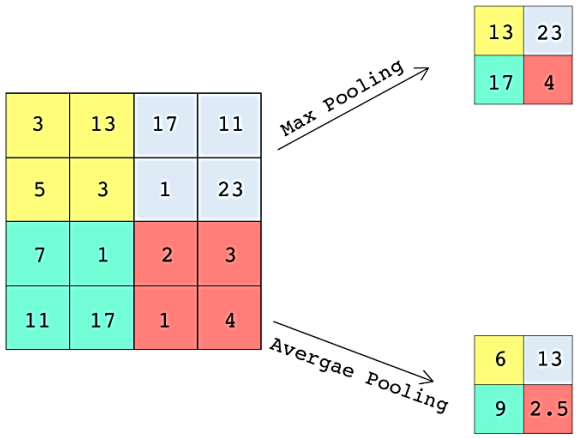


Figure (2.16): Example of (max/average) Pooling operation[67].

Aside from these two types of pooling, another one is called Global Average Pooling (GAVG-Pooling). This type is used to compute the mean output of each FM. This global layer reduces the feature dimension and prevents statistical models from grasping the location information from embedded pixel training data [69].

2.4.5 Absolute Value (ABS) Layer

The ABS layer forces the statistical modeling to consider account the sign symmetry (sign) existing in the noise residuals (NRs). The significance of this layer has been demonstrated in [19].

2.4.6 Batch Normalization (BN) Layer

The BN layer [70, 71] is used to accelerate training, which is a common method in CNN models for image classification. The procedure

includes a mini-batch normalization for each input. The BN layer subtracts the mini-batch mean and divides the result by the mini-batch standard deviation. The usage of a BN successfully eliminates the problems of gradient vanishing and over-fitting. Furthermore, it lowers the likelihood of the network falling to bad local minima in training phase.

Let B=$\{x_{1...m}\}$ a mini-batch, μ_B the mini batch mean, σ_B^2 the mini batch variance, $\hat{x}_i$ the normalized values, ε is a constant for numerical stability, γ is the scale factor , β is the offset and y_i the linear transformations of the input (i.e. the output) which are calculated as shown in the following Equations [71]:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.22}$$

$$\boldsymbol{\sigma_B^2 = \sum_{i=1}^{m}(x_i - \mu_B)^2} \tag{2.23}$$

$$\boldsymbol{\hat{x}_i = \frac{\hat{x}_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}} \tag{2.24}$$

$$y_i = \gamma\, \hat{x}_i + \beta \tag{2.25}$$

Parameters γ and β are learnable parameters that are updated during network training.

2.4.7 Fully-Connected Layers

Fully-connected layers [62] are utilized to generate class scores, and then used as network output (i.e., the output layer at the network's end). All of the neurons in this layer are linked to every neuron in the previous layer. Fully-connected layers have the standard layer and hyperparameter settings. It applies transformations to the input data size based on the activations and the parameters (weights and biases of the neurons).

2.4.7.1 Artificial Neuron

Neurons, typically referred to as nodes or units, are the fundamental building elements of a neural network. Figure (2.17) depicts the structure of a neuron. Input data, activation function, and output data are the three major parameters of a neuron. The neuron gets input from one or more sources, which might be other neurons or data fed to it directly. Each neuron multiplies these inputs by weights, then sums the results and feeds the sum to an activation function, which applies some non-linearity to the neuron's output. A neuron's end output is calculated as follows [61]:

$$O = \emptyset(\sum_i (x_i \times w_i) + b), \tag{2.26}$$

where x_i and w_i are the inputs and their weights, respectively, b represents the bias, additional weight in a neuron that is not connected to any neuron or input. When all of the neuron's inputs are zero, it can shift it's output and prevent it from transferring zero to the next layer. After multiplying the inputs by their weights and adding bias, the output is fed into the activation function represented by φ. The output of a single neuron is computed in this procedure.

A neural network is trained by adjusting the weights and biases of all neurons such that the neural network delivers the desired output with the least amount of error.

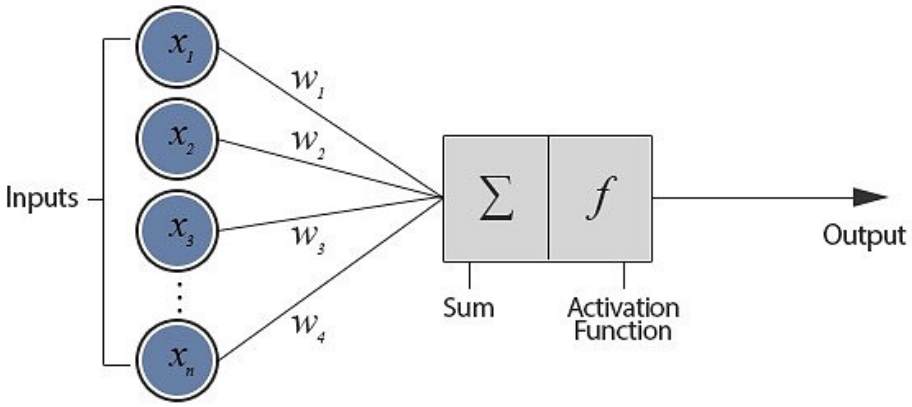


Figure (2.17): Artificial Neuron Structure [39].

2.4.7.2 Architecture of Neural Networks

The neural network [72] is mainly composed of individual neurons arranged in layers, as shown in Figure (2.18). Neurons in each layer are linked to neurons in the previous layer, but neurons in the same layer are not connected together. The first layer is the input layer, the last layer is the output layer, and all the layers in between are referred to as the hidden layers. A neural network can have any number of hidden layers, depending on the related application. Each layer has at least one neuron. The last layer computes the posterior probability of each class label across *K* classes. To determine the network's output, the value of the last hidden layer is given to the output layer.

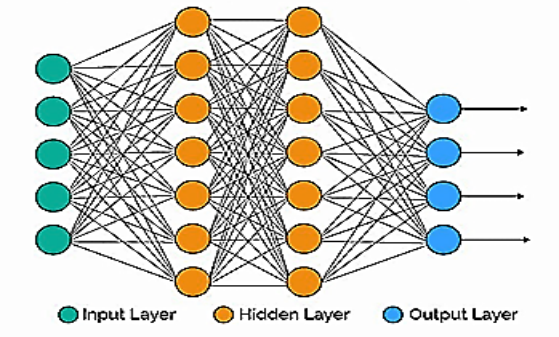
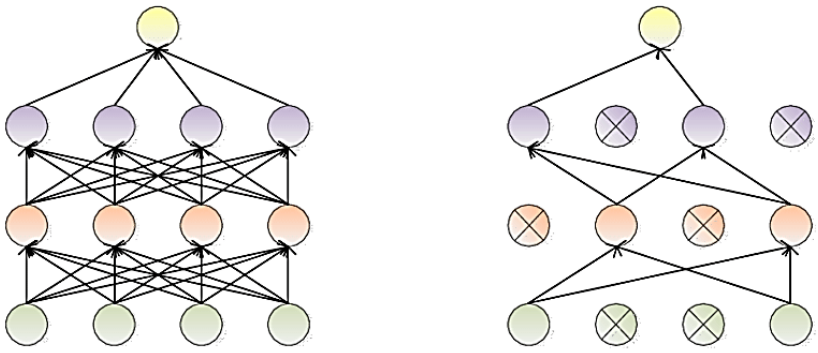


Figure (2.18): An Artificial Neural Network with an Input Layer, two Hidden Layers, and an Output Layer [73].

2.4.7.3 Dropout Layer

Machine learning systems that use deep neural networks with a high number of parameters are extremely powerful. Overfitting, on the other hand, is a major issue in such networks. Large networks are extremely slow to utilize, make it difficult to deal with overfitting by combining the predictions of several different large neural networks at test time. Dropout [74] is a strategy for solving this problem. The fundamental concept is to drop neurons (together with their connections) from the neural network at random with a probability p during training. At the training phase, the particular layer deactivates a percentage of the neurons. This method improved generalization by forcing the layer to deactivate different neurons each time it learned the same thing.

It should be noted that the dropout is deactivated during the test phase, and it provides significant improvements over other regularization approaches. since the dropout is a valuable approach, but the probability p is another hyperparameter that must be cross-validated. Figure (2.19) depicts Artificial Neural Network with and without dropout.



(a) **(b)**

Figure (2.19): Artificial Neural Network (a) With Dropout. (b) Without Dropout[75].

2.4.8 Network Training

The tasks of deep learning involve training the model and putting it to use to make inferences. The training follows the backward path of a network and changes the weights of each layer toward a target regularly. Oppositely, the inference follows the forward path and uses the training weights of each layer to create a prediction based on the input. The network learns to rough the actual to the desired output for recognized input by adjusting the weights across epochs. Gradient backpropagation is the most commonly used method for training neural networks [76].

The gradient descent approach is used to train neural networks in the backpropagation method, where W represents the combined weights vector, that repeatedly updated as follows [61]:

$$W^{(t+1)} = W^{(t)} - \eta \nabla C\left(W^{(t)}\right), \tag{2.27}$$

where the learning rate is represented by η, $W^{(t+1)}$ and $W^{(t)}$ represent the weight vectors at iterations $(t+1)$ and (t), respectively, and $\nabla C(W^{(t)})$ is the cost or error function's gradient concerning the weight vector, W, at iteration (t). The preceding equation can be generalized by $w \in W$, and expressed as follows [61]:

$$w^{(t+1)} = w^{(t)} - \eta \frac{\partial C(w^{(t)})}{\partial w} \tag{2.28}$$

Based on Equation (2.28), it can be observed that the gradient process is based on computing the gradient of the cost function concerning each weight. According to the differentiation chain rule, if $y = f(x)$, and $z=f(y)$, then the following formula is true [61]:

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \tag{2.29}$$

This formula applies to any number of variables. The gradient of the error function controls whether the weight is increased or decreased by the training algorithm. The gradient's sign informs the algorithm of the following:

- Zero gradient: The weight does not affect the network's error.
- Negative gradient: The weight increases to get a smaller error.
- Positive gradient: The weight decreases to get a smaller error.

Figure (2.20) depicts neural network training via backpropagation. The training process, as described in this Figure, includes both a forward and a backward pass. During the first phase of training, an input vector with a known output is propagated forward through the network. The weights values are generated at random or by other techniques. The output layer error values are calculated using a cost function by comparing the network output to the desired output for this input. The output layer's error values propagated backward through the network to determine the error values of the hidden layers.

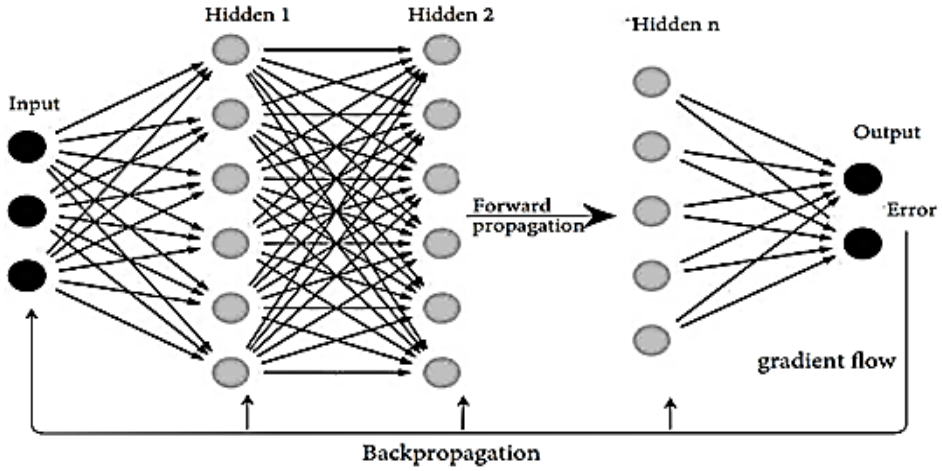


Figure (2.20): Artificial Neural Network Protocol [39].

In the case of image classification, training the convolutional part of CNN will update the model's weights to minimize the difference between the prediction and the ground truth label, while inference utilizes images as input to identify the image class [77]. A four-layer CNN's forward and backward paths are shown in Figure (2.21).

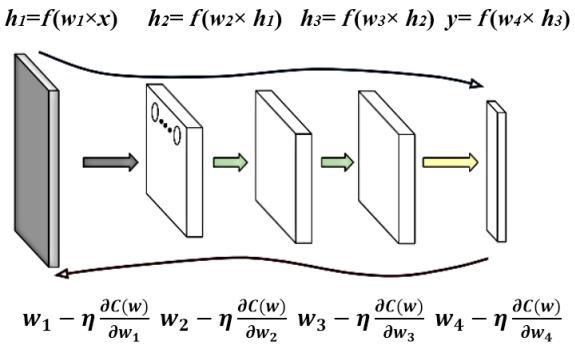


Figure (2.21): The Forward and Backward paths of Four-Layer CNN [76].

The weights of each layer are denoted by $w(\bullet)$ in Figure (2.21). A convolution process between w and x is denoted by $w\times x$. η is the rate of learning. The activation function is denoted by $f(\bullet)$. C represents the cost function to be optimized, which is the difference between the prediction and ground truth labels. The backward path, which is more computationally intensive than the forward path, computes the partial derivatives of the preceding layers' input. In order to complete the partial derivative computation, each layer must save the intermediate results. As a result, the backward path calculation requires more processing and storage than the forward path.

By incorporating momentum, the Adaptive Moment Estimation (Adam) [78] Algorithm is an extension of gradient descent, and a natural successor to Adaptive Gradient (AdaGrad) [79] and Root Mean Squared Propagation (RMSProp) [80] approaches. A moving average of the

current and previous gradients is used to update the parameters W at each update. This helps the cost function to converge rapidly and with fewer oscillations to a local minimum. Adam is a stochastic gradient descent replacement optimization method for training deep learning models. It combines the characteristics of the AdaGrad and RMSProp methods to provide an optimization technique that can handle sparse gradients on noisy problems [78].

2.4.9 Weights Initialization

The weights initialization [81] influence the time necessary to reach the network convergence. It is essential to initialize the network with the appropriate weights in order to prevent layer activation outputs from exploding or vanishing during a forward pass through a deep neural network. When the weights are started with a large value, for example, and the sigmoid is utilized as the activation function, the input data increases rapidly with each passing layer. Because the sigmoid function tends to be flat for considerable input, it eventually becomes so large, and then this makes the sigmoid function becoming useless. As a result, the activation function will be saturated, and the gradients will be close to zero. The effort spent training the network will be wasted if it does not achieve the intended performance. Setting the values of the random weights within a defined range is the most popular but least beneficial technique to weights initialization. Therefore, Glorot and Bengio [82] introduced the technique for generating weights from a distribution with a mean and variance of zero. Weight ($W_{i,j}$) is initialized at each layer using the following frequently used heuristic [81]:

$$W_{i,j} \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_k+n_{k+1}}}, \frac{\sqrt{6}}{\sqrt{n_k+n_{k+1}}}\right], \quad (2.30)$$

where W includes the matrix of weights between layer k and $k+1$, $U[-b, b]$ represents the uniform distribution in the interval ($-b$, b), and n is the number of inputs in layer k.

2.5 Swarm Intelligent

The collective intelligence of decentralized, self-organized systems inspires swarm intelligence. A swarm is a group of interconnected individuals capable of achieving global goals by cooperating on space exploration. The term intelligence is based on groups of interactions between individuals and between individuals and their environment. Individuals in a swarm have a general stochastic (or chaotic) propensity to converge toward a population center of mass on critical dimensions, resulting in convergence on an optimum. [83]

This section introduces basic information on using the swarm intelligence Algorithms as a FS, the standard and binary versions of the PSO Algorithm that related to the proposed work.

2.5.1 Swarm Intelligence as a Feature Selection

The feature extraction approach maps the input data into a new space, resulting in the original dataset being represented in the new space with a lower dimension [57]. This is a common point between image steganalysis and pattern recognition problems. Because the feature vectors of stego-image and cover-image differ, classifying test images can be accomplished by processing the feature vectors of these types of

images; feature components that differ the most in both images are regarded as more influential features [20].

The retrieved features by feature extraction stage might contain an number of irrelevant and redundant features. Therefore, FS techniques are introduced to reduce such noisy features. The feature extraction methods are divided into four sub-categories: filter, wrapper, embedding, and hybrid. Each feature in the class of filter-based techniques is assessed independently based on its discriminative effectiveness across classes. A learning strategy is used in wrapper-based algorithms to examine the effectiveness of feature subsets iteratively [84]. In the embedded category, FS is an integrated element of a model's learning process, and hence the learning and FS parts cannot be separated [85]. The last class of FS approaches is hybrid-based methods, which benefit from the computational efficiency of filter-based methods as well as the suitable performance of wrapper-based methods. PSO is considered one of the most important Algorithms in swarm intelligence techniques.

2.5.2 Standard PSO Algorithm

Kennedy and Eberhart introduced PSO as a SI-based optimization approach in 1995 [86, 87]. It simulates animal social behavior, such as bird folk and fish schooling. In PSO, the swarms of candidate solutions are represented in the search space as particles. It begins with a population of particles being randomly initialized. Particles move through the search space, changing their positions depending on the experiences of their own and neighboring particles in order to find the best solution. During the movement, the current position of particle *i* is represented by a vector $x_i = (x_{i1}, \dots, x_{iD})$, where D represents the search space dimensionality. v_i =

(v_{i1}, v_{i2}, …, v_{iD}) represents the velocity of particle i. The best previous position of a particle, recorded as the personal best, is denoted by *pbest*, while the best position attained by the swarm is denoted by *gbest*. PSO finds the best solution by updating the position and velocity of each particle in accordance with the following Equations [86]:

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{2.31}$$

$$v_{id}^{t+1} = w \times v_{id}^{t} + c_1 \times r_{1i} \times (p_{id} - x_{id}^{t}) + c_2 \times r_{2i} \times (p_{gd} - x_{id}^{t}) \tag{2.32}$$

The t-th iteration of the evolutionary process is denoted by the letter t. The d-th dimension in the search space is denoted by d ϵ D. w is the inertia weight, which included in PSO to stabilize its local and global search capabilities. Its value can reflect changes in particle velocity, impacting the Algorithm's convergence performance [88]. $c1$ and $c2$ are acceleration constants. r_{1i} and r_{2i} are uniformly distributed random values in the range [0, 1]. The components of p_{best} and g_{best} in the d- th dimension are represented by p_{id} and p_{gd}, respectively. As shown in Equation (2.32), v_{id} constrained to [-v_{max}, v_{max}] according to a predetermined maximum velocity, $v_{max,d}$ [89]:

$$v_{id} = \begin{cases} v_{id}, if\ |v_{id}| \le v_{max,d} \\ v_{max,d}, if\ v_{id} > v_{max,d} \\ -v_{max,d}, if\ v_{id} < -v_{max,d} \end{cases} \tag{2.33}$$

The flowchart of basic PSO is shown in Figure (2.22), where each particle is considered to have the whole population as its topological neighbors. First, each particle is given a random velocity and position in a D-dimensional search space. Second, the fitness of each particle is evaluated using a preset fitness function, and the velocity and position of each particle. The method continues with updates each particle's position and velocity values in order to find the optimal solution until a

predetermined stopping condition is reached. A sufficient fitness value or a max number of iterations can be used as the stopping condition.

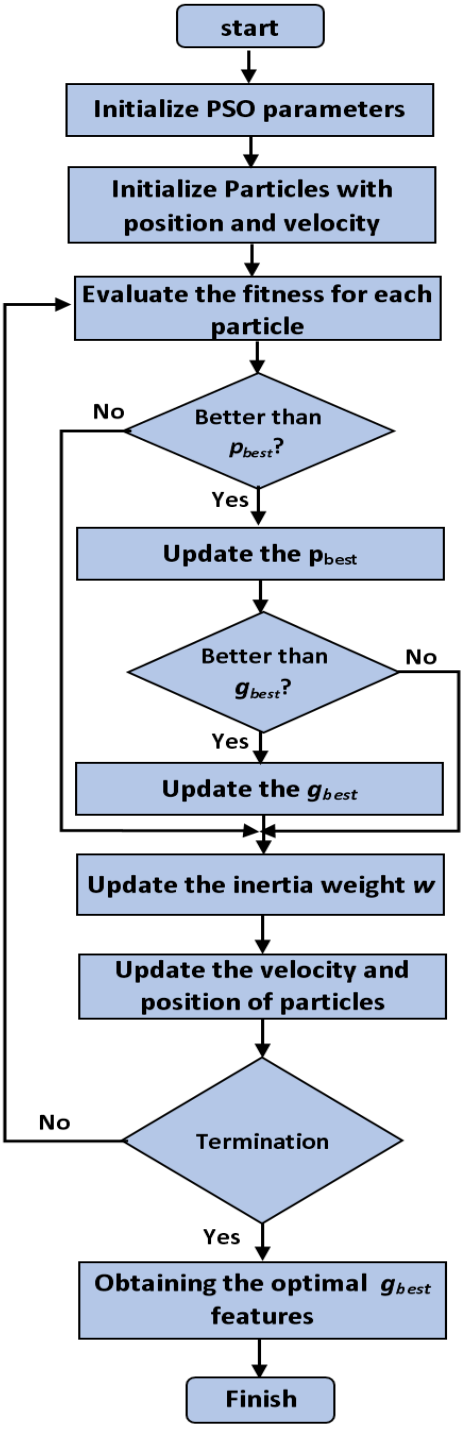


Figure (2.22): The Standard PSO Flowchart [90].

PSO has many advantages such as [91]: easy to implement, simple in concept and few parameters to adjust. So, PSO is applied to solve too large number of applications. PSO, like most techniques of evolutionary algorithms, can be used for solving most problems of optimization. At this time, PSO algorithm is applied to all types problems whether its search space variables are real or binary. There are a major number of problems that its results obtained by PSO are very competitive. One of more important application of PSO is to solve Multi-Objective Optimization Problems (MOOPs) [92, 93, 94] . In MOOPs, there are multiple objectives need to be optimize at the same time. In most MOOPs, no one solution is found to be optimal to all objectives. But, there are a huge of alternative solutions which are known as a Pareto front. In addition, these solutions are equal in their preference. So, there is a difficulty in choosing between them. MOOPs is one of the important application areas which has studied by PSO algorithm. Many of algorithms have been designed to handle MOOPs by using PSO. In the first algorithm, that was used to solve MOOPs, is converted Multi-Objective Optimization Problem to a single objective optimization problem.

2.5.3 Binary PSO (BPSO)

PSO was first presented as an optimization approach for dealing with continuous problems. However, many optimization problems, such as FS, occur in a discrete space with qualitative distinctions between variables and levels of variables. Therefore, Kennedy and Eberhart [95] created a BPSO approach to handle these discrete problems to expand the PSO Algorithm's use. Each particle's position is formulated as a binary representation generated at random; bit values "0" and "1" denote a non-

selected and selected feature, respectively. In BPSO, the velocity indicates the probability of an element in the position having the value "1". The velocity is still updated using Equation (2.32). A sigmoid function $s(v_{id})$ is used to convert v_{id} in the range between 0 and 1, as shown in the following Equation [96]:

$$S(v_{id}) = \frac{1}{(1+e^{-v_{id}})} \tag{2.34}$$

To updates the positions of BPSO for each particle, the following formula is used [95]:

$$x_{id} = \begin{cases} 1, if\ rand(.) < s(v_{id}) \\ 0, \quad otherwise \end{cases}, \tag{2.35}$$

where *rand*(.) returns numbers generated at random from a uniform distribution in the range (0,1).

The inertia weight has a significant impact on the PSO Algorithm's performance because it has the property of updating particle inertial movement and increasing the search space, and also its value can reflect the change in particle velocity. The inertia weight value is strongly related to the searching ability of particles [97]. The greater the inertia weight value, the larger the step size required for a global search, indicating that global optimization is effective but local searching is poor. The lower the value of the inertia weight, the stronger local optimization, but the weaker the global searching ability. If the inertia weight parameter is set too high, the particle swarm may miss the ideal solution, causing the algorithm to fail to converge to the optimal solution. The inertial weight approach described by Yang and Gao in [98] is used in this work, which adaptively adjusts the inertia weight depending on particle velocity information, as indicated in the following Equation [98]:

$$w^{(t+1)} = w_{max} - (w_{max} - w_{min}) \left(\frac{t}{t_{max}}\right)^{\alpha}, \quad (2.36)$$

where w_{max} and w_{min} are the maximum and minimum of inertia weight, respectively, and $\alpha=1/\pi^2$. t and t_{max} represent the current and maximum iteration, respectively. In this proposed BPSO, the best values of w_{max} and w_{min} are determined as recommended by the authors, which are as follows: w_{min} = 0.4 and w_{max} = 0.9.

2.6 Classical Machine Learning Algorithms

Classical machine learning techniques can be used to model the selected features. This section discusses all of the Algorithms used to model these features in this dissertation.

2.6.1 SVM Classifier

SVM has supervised learning models that aim to identify the optimum hyperplane to split two different classes of training data in order to create the best model for test data. Vapnik [99] proposed the SVM technique, which demonstrated extremely high accuracy for image feature classification [100]. It allows to selecting kernels such as linear, radial basis function, and polynomial. The hyperplane learning procedure is used to model utilizing the linear kernel by converting the problem using some linear algebra. The dot product is used to construct the prediction between the input (x) and each support vector (x_i) for a new input as represented in the following Equation [99]:

$$f(x) = \text{sum}\,(w_i \times (x, x_i)) + \beta\,, \quad (2.37)$$

where w_i is the weight vector and β denotes the bias. The inner products of a new input vector (x) with all support vectors in training data are

included in Equation (2.37). The learning algorithm must estimate the training feature vectors' coefficients w_i and β (for each input).

2.6.2 LR Classifier

LR [101-103] is a common approach for statistical modeling in which the probability, *P*, of a dichotomous outcome event is concerned with a number of explanatory factors in the form [101, 102]:

$$logit(p) = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n = \beta_0 + \sum_{i=1}^{n} \beta_i x_i \quad (2.38)$$

In Equation (2.38), β_0 is the intercept, while β_1, β_2, ..., β_n are represent the measurements concerned with the explanatory variable x_1, x_2, ..., x_n. A dichotomous variable has just two values, yes/no, on/off, or 1/0, and is used to describe the event occurrence or non-occurrence. In general, LR imposes less strict constraints since it does not need Gaussian distributed independent variables and does not require linearity between the explanatory and response variables. The probability of an event occurring as a function of the explanatory variables is nonlinear, as shown by following Equation [102]:

$$P(x) = \frac{1}{1+e^{-logit(P(x))}} = \frac{1}{1+e^{-(\beta_0+\sum_{i=1}^{n}\beta_i x_i)}} \quad (2.39)$$

In this Equation:

- $P(x)$ is the estimated probability of falling to the default class. In binary classification, the default class is marked with a 1, and the opposite class is marked with a 0. $P(x)$ expresses the probability of an example falling to the default class on a scale between 0 and 1.
- $(1 + e^{-z})$ is the sigmoid function.
- $\beta_0 + \sum_{i=1}^{n} \beta_i x_i$ is the linear model within LR.

2.6.3 Naïve Bayes (NB) Classifier

The NB classifier [104] is based on the Bayesian theorem with the naïve assumption of independence between each pair of features. Despite apparently oversimplified assumptions, this classifier has performed quite well in many applications in the real-world. It takes a minimum training time compared to conventional supervised or unsupervised learning algorithms. To illustrate the NB classifier, let *B* be a data tuple. *Pr*(*A/B*) is the posterior probability of *A* conditioned on *B*. In contrast, *Pr*(*A*) is the prior probability of *A*. Similarly, *Pr*(B|A) is the posterior probability of *B* conditioned on *A*. *Pr*(*B*) is the prior probability of *B*. Bayes' theorem is serve as a method of calculating the posterior probability, *Pr*(A|B), from *Pr(A)*, *Pr*(B|A), and *Pr*(*B*), as described in the following Equation [105]:

$$Pr(A|B) = \frac{Pr(B|A)\ Pr(A)}{Pr(B)} \tag{2.40}$$

2.6.4 Random Forest (RF) Classifier

The RF is an ensemble of Decision Trees (DT) that are mainly trained using the "bagging" approach [106], as shown in Figure (2.23). RF actually improves bagging by considering only a small subset of features at each split of each tree rather than the entire set.

Given *n* available features, a subset will have $\sqrt{n}$ features chosen at random. As a result, the algorithm decorrelates each used tree using this described technique. Each tree makes a decision (class), and the class with the most votes represents algorithm's prediction.

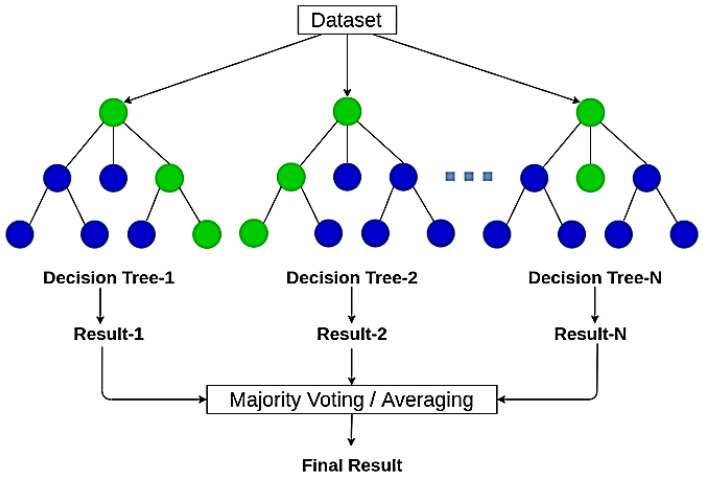


Figure (2.23): A typical Random Forest [107].

RF is a very useful and quick method with good accuracy even with a default setting of hyperparameters and is resistant to over-fitting [106]. The sole constraint is the amount of trees; the more trees there are, the more accurate getting but slower the algorithm becomes.

2.7 Evaluation Metrics

To evaluate the modeling algorithms' performance, an appropriate metric must be chosen. The number of test features that the trained classifier correctly and incorrectly predicted is used to evaluate the classification model. These counts are tabulated in a table known as a confusion matrix. Table (2.1) shows the confusion matrix for a binary classification problem and can be extended to any number of classes [108].

Table (2.1): Confusion Matrix for 2-class Problem

Actual Prediction	Predicted Class	
	Class = "Stego"	Class = "Cover"
Class = "Stego"	True Positive (T_P)	False Negative (F_N)
Class = "Cover"	False Positive(F_P)	True Negative (T_N)

The confusion matrix contains essential aspects of the modeling results; these are:

• **True Positives (T_P):** Represents the number of positive target class observations successfully classified by the model.

• **False Negatives (F_N):** Represents the number of observations in the positive target class that the model incorrectly identified as being in the negative target class.

• **False Positives (F_P):** Represents the number of observations in the negative target class that was incorrectly identified as being in the positive target class by the model.

• **True Negatives (T_N):** Represents the number of data properly identified by the model as being in the negative target class.

Multiple metrics can be concluded from the confusion matrix, these are [109]:

- **Accuracy :** Represents the total number of correct predictions made by the model.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.41}$$

- **Recall:** It is the proportion of relevant positive classes identified correctly by the classifier.

$$Recall = \frac{TP}{TP+FN} \tag{2.42}$$

- **Precision:** It is the proportion of positive classes correctly identified considering all the positive classes predicted.

$$Precision = \frac{TP}{TP+FP} \tag{2.43}$$

While a confusion matrix provides the information needed to determine how well a classification model performs, comparing the output

of different classifier models would be more convenient by summarizing this information with a single value. This can be done using F-measure that considers a common way of evaluating the accuracy of the Machine Learning experiments [110]. The F-measure is the harmonic mean of the precision and recall metrics. This metric allows for a balanced expression of the model's recall and precision, as described in the following Equation [111]:

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (2.44)$$

CHAPTER THREE

PROPOSED SYSTEM DESIGN

CHAPTER THREE

PROPOSED SYSTEM DESIGN

3.1 Introduction

This chapter introduces the detailed design and implementation of the proposed system for blind steganalysis of content-adaptive image steganography based on CNN and FS. The proposed system has been partitioned into many stages, which are: data preparation using five content-adaptive stego algorithms, pre-processing using NRF and normalization, datasets splitting into training and testing, feature extraction using CNN, FS using BPSO, and finally, the classification stage using SVM classifier to distinguish whether the image contains hidden information based on the best features selected.

3.2 Proposed System Design

This section introduces the implementation-related techniques and methods. First, the datasets for image steganalysis purposes are prepared. Then, pre-processing on images is described that includes applying NRF and data normalization. Next, the datasets are splitted into training and testing sets. After that, the proposed architecture of CNN-based feature extraction and the BPSO algorithm for FS are introduced. Finally, the SVM classifier has been learned using training selected feature vectors and using it for prediction to discriminate between images with secrete information from that the cover. Figure (3.1) shows the general methodology design of the proposed system.

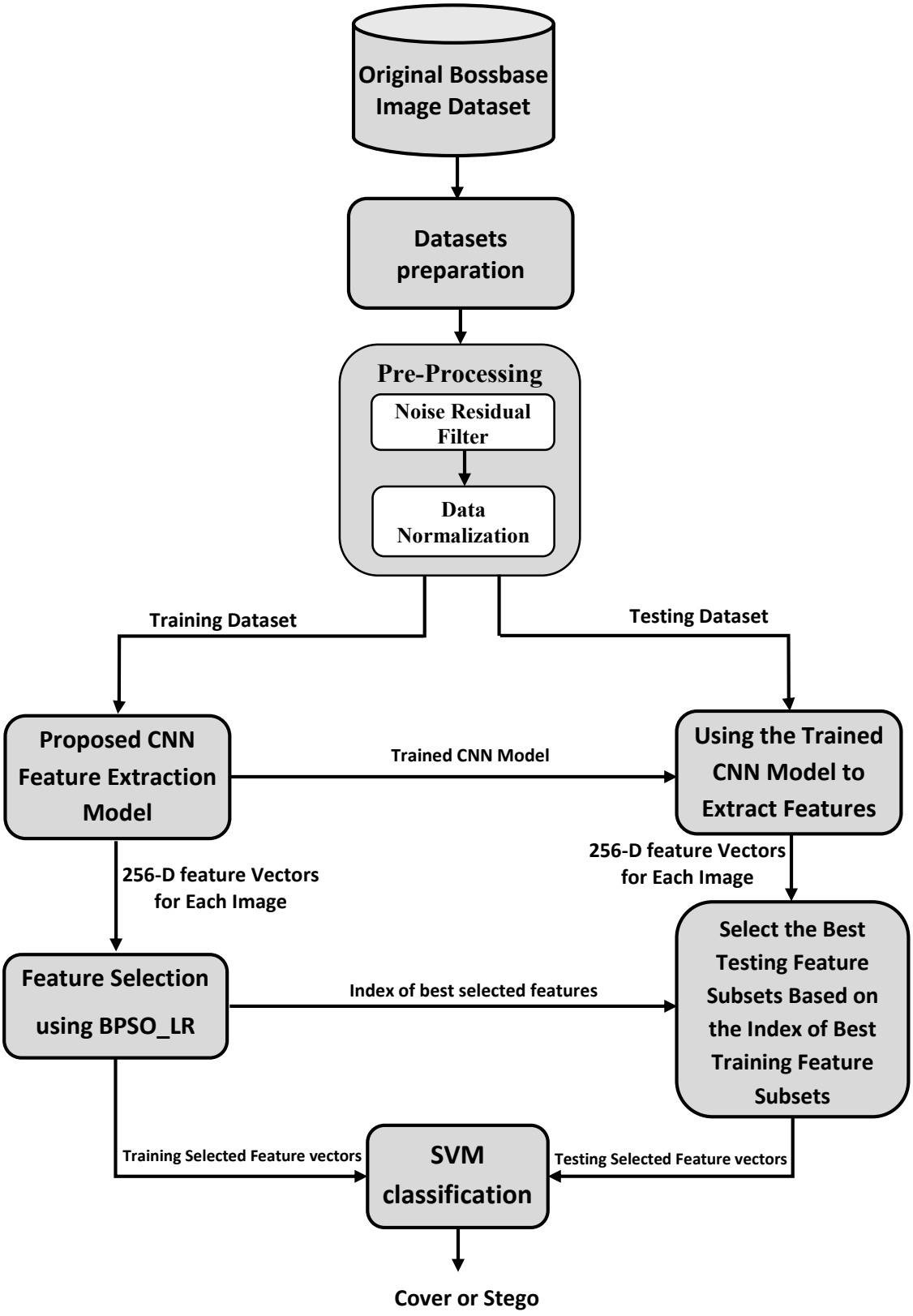


Figure (3.1): The General Methodology Design of the Proposed System.

The general design of the proposed method involves the following stages:

- Datasets Preparation Stage.
- Datasets Pre-processing Stage.
- Datasets Splitting Stage.
- Features Extraction Stage.
- Feature Selection Stage.
- Classification Stage.

3.2.1 Datasets Preparation Stage

Due to the unavailability of a publicly available datasets including images embedded with various steganographic methods, extra efforts and time are needed to prepare these datasets using many different steganographic algorithms at different insertion rates. This step is essential since it determines the results of the experimentation and evaluation.

The BOSSbase v1.01 dataset was utilized to prepare the steganography datasets, which were never compressed and originated from seven different cameras. The sample images from this dataset are depicted in Figure (3.2). The BOSSbase is frequently used for analyzing steganography and steganalysis experiments. It includes grayscale PGM files and is appropriate for spatial-domain applications. It was also recognized by the simplicity of the PGM format, which is the lowest common denominator grayscale file format and was designed to be simple to understand. A PGM file is a grayscale image file saved in the portable gray map (PGM) format and encoded with one or two bytes (8 or 16 bits) per pixel. It contains header information, that defines the PGM format type ("P2" for text or "P5" for binary, and a grid of numbers that represent

different shades of gray from black (0) to white (up to 65,536). The header of a PGM image file consists of:

- First line containing the *signature* of the image file and identifies the file as PGM
- Second line is the *comment line*
- Third line provides information about the *number and rows and columns* of data stored in the file, and
- Fourth line *specifies maximum gray level* contained in the image.

Figure (3.2): BOSSbase Sample Images. Selection out of total 10,000 Grayscale PGM Images with 512x512 size.

Data follows the header information and is written in pixel values (in *text or binary* format). Data is in raster order, which indicates that all data for the first row of the image is written first, then for the second row, and so on. The origin of the coordinate system for a PGM image is located on the top left corner.

The BOSSbase contain a total of 10,000 images with a size of 512×512 pixels. To speed up the computing power, these images are scaled into 256×256 pixels in all experiments. Also, the experiments were performed using five content-adaptive steganographic algorithms that adaptively being hide the information in the spatial domain. These algorithms are HUGO, WOW, S-UNIWARD, HILL, and MiPOD. The original BOSSbase images are embedded with hidden information at the rates of 10%, 20%, 30%, and 40% bpp.

Accordingly, after applying steganographic algorithms, four collections of datasets become available for each steganographic algorithm, including 10,000 stego images for each embedding rate in addition to the original cover images. So, as a result, the final total of (5×4×10,000=200,000) stego images, where 5 represents the number of the steganographic algorithm, 4 represents the number of embedding

rates, and 10,000 represents the number of images used for embedding at each steganographic algorithm.

The new datasets are created with a total of 200,000 images, which were distributed as follows:

- 40,000 images embedded with HUGO steganographic algorithm with the rates of 10%, 20%, 30%, and 40% bpp.

- 40,000 images embedded with WOW steganographic algorithm with the rates of 10%, 20%, 30%, and 40% bpp.
- 40,000 images embedded with S-UNIWARD steganographic algorithm with the rates of 10%, 20%, 30%, and 40% bpp.
- 40,000 images embedded with HILL steganographic algorithm with the rates of 10%, 20%, 30%, and 40% bpp.
- 40,000 images embedded with MiPOD steganographic algorithm with the rates of 10%, 20%, 30%, and 40% bpp.

We named these datasets at Babylon University Faculty of Information Technology. Because this dataset is not available online, it is possible to uploaded on the college website so that image steganalysis researchers can use it to reduce their efforts in preparing these dataset.

3.2.2 Pre-processing Stage

This stage includes two steps, applying NRF on images and normalized the filtered images.

3.2.2.1 Applying Noise Residual Filter

This step includes convolution of the image with a predefined filter to suppress the image contents, which has proved very efficient to increase SNR through exposing the stego NRs. Because this fixed kernel is not trainable, it cannot be learned from the training data when updating the weights using the backpropagation method. For image steganalysis, certain CNN architectures utilize trainable kernels that are initialized with high-pass filter kernels. Because the kernel parameters/weights are changed throughout the training process and some characteristics (e.g.,

zero-sum) are no longer guaranteed, these trainable kernels are unlikely to continue performing high-pass filtering. As a result, this filter will lose the properties for which it was designed.

In this stage, it is adapted to use a single filter designed in the SRM method. The 5×5 NRF is defined as follows:

$$F = \begin{bmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{bmatrix} \tag{3.1}$$

Generally, this filter is considered as simulating the feature extraction process, and it can contribute to improve the performance of the proposed CNN, as will be demonstrated in Chapter 4 experiments. This filter is considered a symmetric filter and has a gradient characteristics causes the local variation to be highlighted. Figure (3.3) illustrates the applying of the NRF pre-processing step.

In content-adaptive schemes, the secrete information is embedded in regions of images that are not smooth, such as lines. As a result, stego signals are generally thought to be hidden in image noise. Therefore, applying NRFs is likely to enhance the stego information, which is highly useful for image steganalysis. Figure (3.4) illustrates an example of the convolution method in which the output can be calculated for the value $b_{5,5}$ after applying NRF on the image.

Algorithm (3.1) shows the overall steps to apply the NRF. In this algorithm, there are image datasets that represent the cover and five different steganographic methods so that each image in these datasets is

convolved with 5×5 NRF. The result represents the NRs of this convolved image.

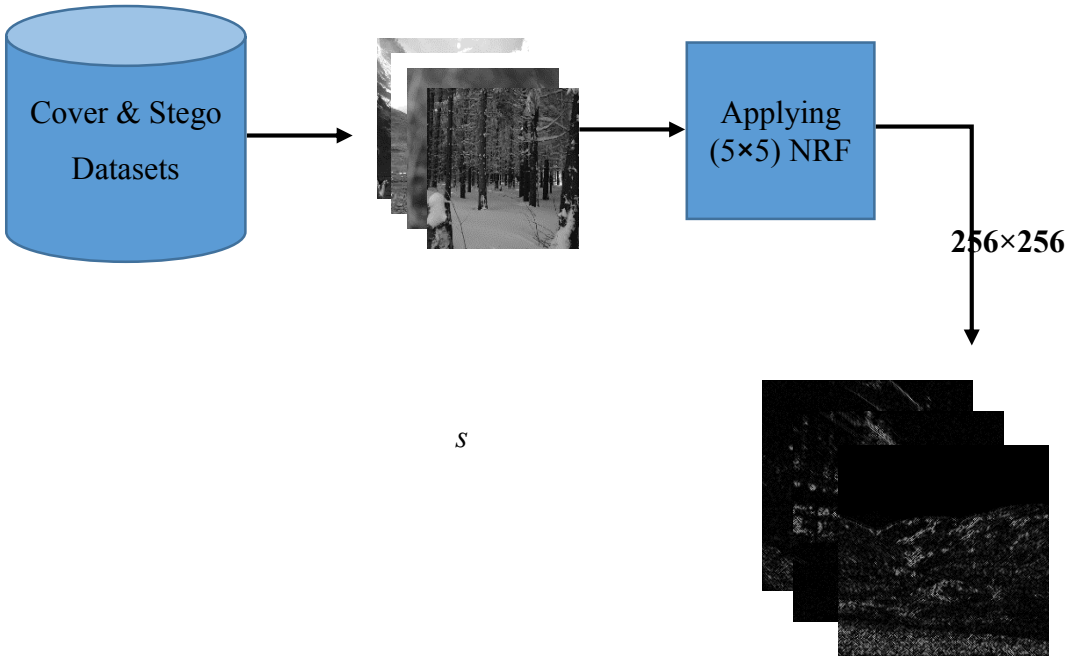


Figure (3.3): Applying NRF Pre-Processing Step.

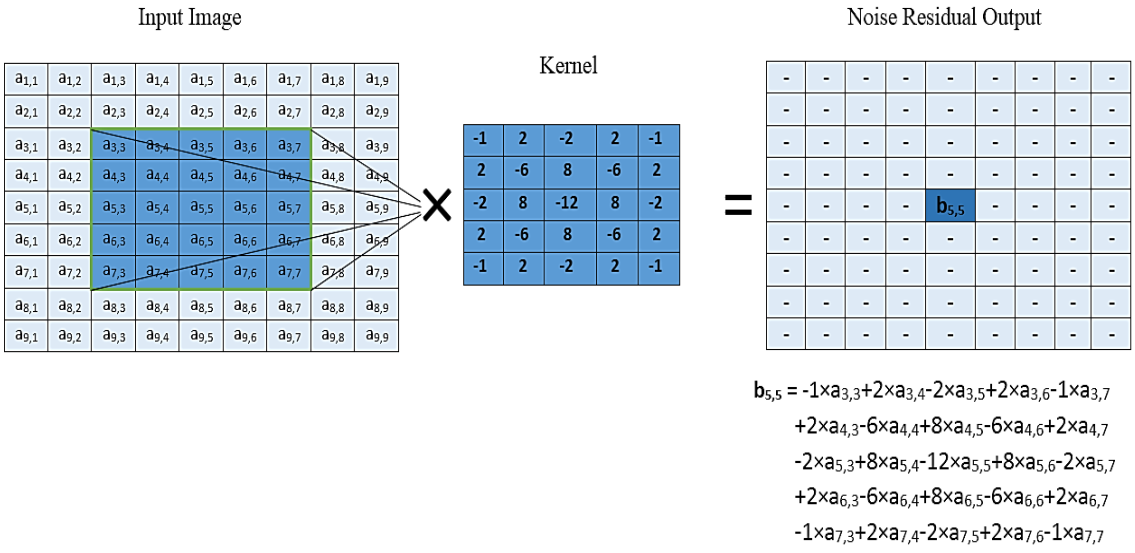


Figure (3.4): Convolution Process of NRF.

Algorithm (3.1): NRF Pre-processing Stage.

Input : Cover Image Dataset D_c, Stego Image Dataset D_i for the ith steganographic Algorithm, the Noise Residual kernel K.

Output: Noise Residuals *NRs*.

1- Begin

2- For each Dataset D_c ***or*** D_i

2-1 For each *Image I in* D_c ***or*** D_i

2-1-1 Convolve I with K to get NRs response

$$NRs = I * K, I \epsilon \{Cover, Stego\}$$

2-1-2 Save *NRs* into pre-processing Image Folder

2-2 End For

3- End For

4- End

3.2.2.2 Data Normalization

To ensure that each input has a uniform data distribution, a data normalization stage is used, and it has been applied for each NR image. Data normalization leads to faster convergence while training the model. To carry out data normalization, the mean of the pixel values and the standard deviation are calculated. Then, the new pixel values of each image are then calculated as subtract each pixel value from the mean and

divided by the standard deviation. After this process, new values of all pixels have a "0" mean and standard deviation of "1".

3.2.3 Dataset Splitting Stage

After completing a pre-processing stage, the procedure includes dividing each dataset into two subgroups: The first group, known as the training, is utilized to learn the model. The second one, known as the testing, is fed into the model for prediction as new unpredicted data. Each dataset is splitted into 70% training and 30% testing.

3.2.4 Feature Extraction Stage

After pre-processing the images with NRF and normalized its results, a feature extraction stage has been applied by the proposed new design of CNN. It is a well known that the CNN performs two tasks at the same time, namely, feature extraction (or feature learning) and classifying the extracted features.

In this dissertation, a new design of CNN architecture is proposed to extract features for blind image steganalysis purposes. The proposed network accepts the normalized NRs as input that resulted from filtering the images, and the final result is a feature vector with 256 dimensions (256-D) that satisfies the best classification by the classification module of CNN. The proposed architecture is initialized by a convolutional (or feature learning) module and ends with a classification module. The next subsections illustrate these two modules in detail.

3.2.4.1 Convolutional Module

The overall architecture of the proposed CNN-based feature extraction is shown in Figure (3.5). The proposed convolutional module is composed of seven base blocks, and each block includes many layers of different purposes. The overall structure can be described as follows:

- Each designed block begins with a BN layer (except base block_1 that proceeded with normalization process) to enforces inputs to have the same distribution and decrease the probability of CNN in the training phase falling into a poor area around local minimum at the gradient-descent process and ending with a pooling layer concerned with down-sampling the learned FMs.
- Different layers are found between BN and pooling layers, such as the convolution, ABS, and activation layers.
- To achieve the diversity of the extracted features, two subnets, that are repeatedly combined and separated, are proposed to be included in the CNN, and different kernel sizes (e.g., 5×5, 3×3, and 1×1) are utilized. Algorithm (3.2) illustrates the overall steps of the parallel subnets included in the proposed CNN. As shown in this algorithm, each block consists of two subnets; it has similar included layers, except that the base block_1 goes beyond the BN layer because the normalization process was previously applied. In the proposed CNN, subnets are included in both base block_1 and base block_4. The applying BN (in step 2-1) on NRs or FM is due to the input in the base block_1 is NRs, while in the base block_4 it is FM. Also, the ABS layer (in steps 2-3) is contained only in both the two sub-nets of base block_1.

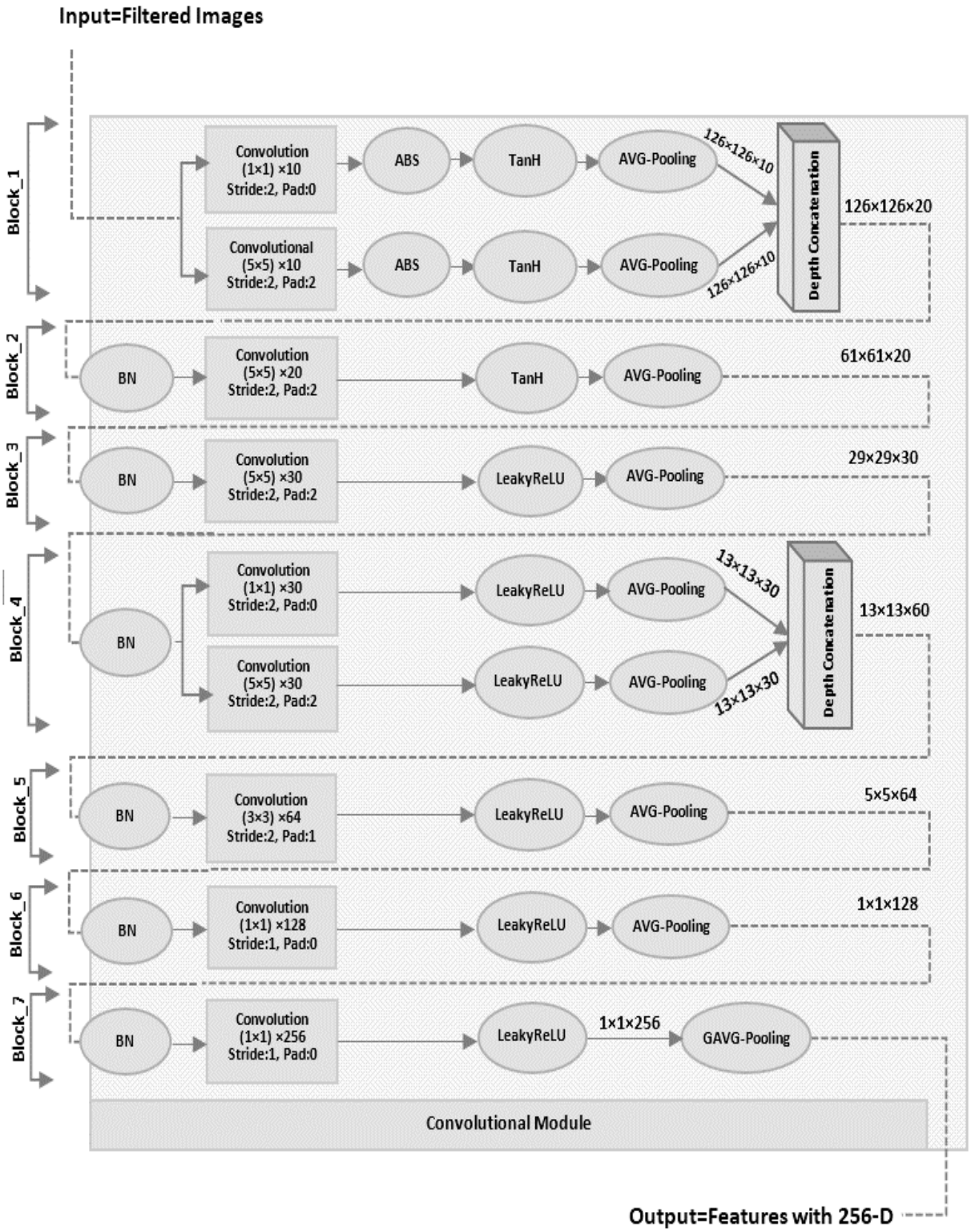


Figure (3.5): Architecture of the Proposed CNN-based Feature Extraction.

Algorithm (3.2): Parallel CNN Subnets.

Input : Noise residuals NR or FM

Output : Parallel Training FM (PTFM)

1- Begin

2- For each subnet **in** block

2-1 BN ← Batch_Normalization(NRs or FM)

2-2 con ← Convolution2D (BN)

2-3 ABS ← Absolute_Value (con)

2-4 act_fun ← Apply Activation_Function (ABS) // Activation //Function ϵ {TanH, LReLU}

2-5 pool ← pooling(act) // pooling ϵ {AVG, GAVG}

3- End For

4- Depth Concatenation of the two subnets

5- End

- The use of small-size convolutional kernels has the benefit of capturing different local correlations among image pixels and thus extracting useful features for image steganalysis. For accurate detection, a large kernel can combine sufficiently weak stego signals into a stronger one. In the base block_1 to base block_4, including only one subnet of separated subnets, (5×5) kernel size is used for the convolution layer. To limit statistical modeling, the spatial size of the kernel is reduced into (3×3) size in the base blofck_5 and then into (1×1) in both last two base blocks. Each convolution filter is initialized using Glorot (Xavier) uniform distribution (Eq. 2.30) with a stride=1 and zero-initialized biases.

In the first block, bias learning has been disabled to make FMs symmetric with respect to zero (sign-symmetry).

- For activation functions, in order to present non-linearity to the proposed CNN, the first and second base blocks (included the two subnets) are equipped with the TanH activation function (Eq. 2.18). The first two blocks employ the TanH activation function to limit the range of data values and prevent the deeper layers from modeling larger values. Moreover, using the TanH activation function with the BN layer will ensure that the initial TanH input falls within the quasi-linear area, and hence the gradient backpropagation would not have been trapped at the poor area around local minima. For the other five blocks, the LReLU (Eq. 2.20), with α=0.2, is used where the ReLU activation function and its versions are the most used activation functions for accelerating the network convergence.

- The pooling layer represents the final layer for each designed block, which continuously decreases the size of the FM. The AVG-Pooling layer, which is included in the first six blocks of CNN, lowers the size of the FM by taking into account all the features in the sliding window (within the kernel size) for each FM. In the final block, the GAVG-Pooling layer also reduces the size of FMs by lowering each FM into a single value. It is used to reduce the dimensionality of features (256 FMs of size 1×1 to 256-D features). As a result, the CNN is avoided from grasping the location information of embedded pixels in the training data.

The classification module next learns the classifier and performs the classification process using the resultant 256-D feature vector.

3.2.4.2 Classification Module

The extracted features from the designed CNN blocks are passed to the classification module that comprises three fully-connected (Dense) layers, which are necessary to add additional non-linearity within the proposed CNN in the hierarchical form of feature extraction. The structure of this module is described as follows:

- The first and second layers have 256 neurons each, followed by a "dropout" technique with a probability of 0.5.
- The benefit of using "dropout" technique is to reduce the effect of over-fitting, where this network do the same think in different connections.
- The last fully-connected layer includes just two neurons, equal to the number of output classes of the network.
- On the final fully connected layer, a softmax activation function (Eq. 2.21) is used to standardize the network's output between [0; 1], and these two scores indicate the probability of falling to the cover or stego class

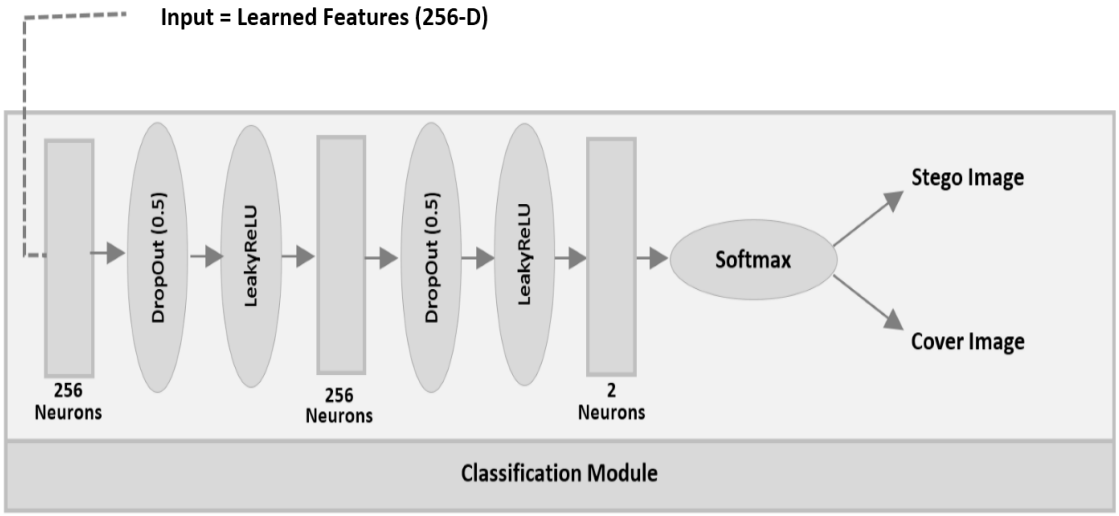


Figure (3.6): Architecture of the Classification Module of the Proposed CNN.

labels. The class with the highest score is chosen as the final class label. Figure (3.6) illustrates the overall steps CNN classification module.

Also, algorithm (3.3) illustrates the classification modules that includes the fully connected layers; the inputs for this layer are a vector with 256 features. Moreover, the Glorot (Xavier) uniform initializes the weights in the fully connected layers. The weighted summation for each neuron in each fully connected layer must be subjected to the activation function. LReLU activation function is also used for the fully connected layers with the scale value α equal to 0.2, in the classification stage.

Algorithm (3.3): Classification Module.

Input: Vector of 256-D features, no. of hidden layers *n*.

Output: Class Labels.

1- Begin

2- Weights Initialization W_i.

3- **For** *i* **from** *1* **to** *n*

3-1 $net = 0$

3-2 **For** *j* **from** *1* **to** *d* // *d* is the number of inputs

3-2-1 $net \leftarrow net + W_{i,j} * x_j$

3-3 **End for**

3-4 $y_i = \begin{cases} net, & net \geq 0 \\ \propto * net, & net < 0 \end{cases}$ // LReLU activation function (Eq. 2.20)

4- **End for**

5- Apply softmax Function (Eq. 2.21)

6- End

This stage is a neural network part that is considered a classification part for CNN through which can obtain the best feature collection.

To demonstrates the whole feed-forward and feed-backward propagation steps for training the proposed CNN-based feature extraction, algorithm (3.4) shows these detailed steps.

Algorithm (3.4): Learning the Proposed CNN for Feature Extraction.

Input : Noise Residual Images *R*, The untrained CNN model *M*, and learning rate.

Output : a well-trained CNN model.

1- **Begin**

2- **Repeat**

Forward Propagation:

2-1 **Repeat**

2-1-1 BN ← Batch_Normalization(FM)// FM is the Feature
// Map resulted from
// the previous block

2-1-2 con ← Convolution2D(R **or** BN)

2-1-3 ABS ← Absolute_Value(con)

2-1-4 act ← Apply Activation_Function (ABS **or** con)
//Activation Function
//ϵ {TanH, LReLU}

2-1-5 pool ← pooling(act) // pooling ϵ {AVG, GAVG}

2-2 **Until** end_of_blocks

To be continue

2-3 fc_1 ← fully_connected(pool)
2-4 drop_1 ← DropOut(fc_1)
2-5 fc_2 ← fully_connected(drop_1)
2-6 drop_2 ← DropOut(fc_2)
2-7 class_label ← SoftMax(fc_2) // Eq. (2.21)
Backward Propagation:
2-8 conduct backward propagation with "Adam" optimizer
3- **Until** *M* convergence
4- Use the trained model to predict the class label
5- **End**

The step (2-1-1) in the algorithm has applied to all base blocks except base block_1. Also, step (2-1-3) has been included only after the convolution layer in the base block_1 of the proposed CNN. The steps (2-3 to 2-7) represent the three fully connected layers. The first and second fully connected layers include 256 neurons followed by a dropout layer with a probability of 0.5. The last layer represents the softmax activation function score to discriminate between the cover and stego image.

3.2.5 Feature Selection Stage

A FS technique is employed at this stage to increase classification performance in deep learning models. It is considered one of the main contributions, and its methods are used to select the most informative feature combination among the features derived from the proposed model's CNN layers with the minimum information loss for the blind image steganalysis problem. In other words, it tries to construct a subset by selecting better features from the existing feature set.

One of the main goals of this dissertation is to investigate and improve the capability of BPSO for FS. Many studies have shown that BPSO is an efficient search technology choice. In this dissertation,

- The feature subset selection objective includes two folds: decrease the dimensionality of the features and, at the same time, increase the classification accuracy, and hence, increase the performance. Two contradictory objectives are generally taken care by using multi-objective BPSO. As mentioned in the feature extraction stage, the proposed CNN is utilized to generate a feature vector with 256-D elements.

- To applying the FS stage, the learned features by the convolutional part of CNN that satisfied the best classification are extracted from the last layer of the convolutional part.

- Then, a method to select the best discriminative features is proposed by combining the BPSO algorithm with Logistic Regression (LR) classifier. In this method, the fitness function of the BPSO algorithm is the classification accuracy of a LR classifier over training set samples.

- The classification is evaluated by using the accuracy of 10-fold cross-validation. Equation (2.41) is used as a fitness function mentioned in chapter two.

Algorithm (3.5) illustrates the overall steps for feature subset selection by using the proposed BPSO_LR method. In this algorithm, the particle initialization is represented as sequence binary values (*Dimentionality*), where *"Dimentionality"* represents the entire number of features (256-D). After calculating the velocity, the resulted value is transformed into the range "0" and "1" binary values by using the sigmoid function according to Equation (2.34). The sigmoid function calculates the existence of a

specific feature in a feature set. When the v_{id} value is larger than a number between "0" and "1" that randomly generated, it is set to "1", indicating that this feature is taken for the new generation; otherwise, it is set to "0", indicating that this feature is not taken for the new generation. For example, a specific FS approach selects the best features from the 15 features to accomplish a certain steganalysis task. The input feature set may include the features $x_1, x_2, x_3 \ldots, x_{14}, x_{15}$. Now, to perform an optimize for these features by using the proposed BPSO, a particle after any generation might look similar as (0,1,0,1,1,0,0,1,1,0,0,1,0,1,1). In this example, "1" and "0" represent a selected and non-selected feature, respectively. The features chosen by this particle are $x_2, x_4, x_5, x_8, x_9, x_{12}, x_{14}$, and x_{15}, based on the location of 1's and 0's within that particle. Because of the effect of the other particle's g_{best} and p_{best} positions, the "0" and "1" values of this particle may change position in the next generation, implying that another set of features among these 15 may be chosen. The LR fitness measure is used to update a particle's dimension.

Algorithm (3.5): The proposed BPSO_LR for Feature Selection.

Input: No. of particles (*Population_Size*), and Feature set that extracted from CNN on Training data (*Dimentionality*).

Output: Best selected features g_{best} , positions of g_{best}

1- Begin

2- Random Initialization of each particle's position and velocity

3- Repeat

3-1 For *i=1* **to** *Population_Size* **do**

To be continue

3-1-1 Separate Train and Test data using 10-fold cross-validation

3-1-2 Call LR classifier to calculate the fitness function for each particle

3-1-3 Evaluate the fitness for each particle

3-1-4 **If** *fitness of particle i is better than that of* p_{best} **then**

3-1-4-*i* update the p_{best} and save their selected features and indexes

3-1-5 **End if**

3-1-6 **If** *fitness of* p_{best} *of any Particle i is better than that of* g_{best} **then**

3-1-6-*i* update the g_{best} and save their selected features and indexes

3-1-7 **End if**

3-2 **End for**

3-3 **For** *i=1* **to** *Population_Size* **do**

3-3-1 **For** *d=1* **to** *Dimensionality* **do**

3-3-1-i update the velocity according to the Eqs. (2.32 and 2.33)

3-3-1-*ii* update the position according to the Eqs. (2.34 and 2.35)

3-3-2 **End for**

3-3-3 update the inertia weight according to the Eq. (2.36)

3-4 **End for**

3-5 **Until** *Maximum_iteration is reached*

3-6 End

The BPSO algorithm with the LR classifier is used to select the best feature subset and eventually improve the detection accuracy of the classifier.

The main problem of swarm intelligence-based algorithms is balancing the exploration and exploitation levels by adjusting the inertia weight. It is critical to note that a high inertia weight raises the exploration level and enables global search. In contrast, a lower inertia weight increases the exploitation level and allows the algorithm to search locally. However, the inertial weight strategy proposed by Yang and Gao is employed to be used in the proposed BPSO_LR algorithm, which adaptively adjusts the inertia weight based on particle's velocity information, as shown in Equation (2.36).

3.2.6 Classification Stage

After FS operations, the best selected informative features and their index have been gained. Next, a classification stage comes, which is the final stage of the proposed method for steganalysis of blind content-adaptive image steganography. To complete this task, it is important to learn the classifier with the selected feature vectors of the training set that resulted from the FS stage. To evaluate the proposed method, it is proposed to use a SVM classifier that given a best accuracy of the proposed method.

After learning the SVM classifier with the training images features to build a supervised learning model, the same trained classifier is used to predict categorical labels ("Cover" or "Stego" image) of the feature vectors for the testing set that is considered as new unpredicted image features. The SVM classifier is trained on the selected training features,

using the proposed BPSO_LR method, as a training set and the testing performed on the selected testing features its position corresponding to the index of the selected training features. To clarify the steps involved in handling the classification stage, algorithm (3.6) explains the steps in detail.

Algorithm (3.6): Classification Stage of the proposed Method.

Input: Selected Training Feature Vectors ***LSTFV*** & Index ***I*** of g_{best}, Testing Set Feature Vectors ***TSFV***

Output: Classification Accuracy

1- Begin

2- For each Steganographic Algorithm Dataset **do**

2-1 Train the SVM with the features in ***LSTFV***

2-2 Selected the Testing Feature Vectors T_f from ***TSFV*** their index correspond to ***I***

2-3 Evaluate the Trained SVM with T_f by using Eq. (2.44)

3- End For

4- End

CHAPTER FOUR

EXPERIMENTAL RESULTS

CHAPTER FOUR
EXPERIMENTAL RESULTS

4.1 Introduction

This chapter explains the experimental results of the proposed method for steganalysis of content-adaptive image steganography. The experiments are performed on each stage and subtasks. The procedure of experiments begins with a datasets preparation, then applying the pre-processing stage, which involves applying both a NRF. After that, applying the proposed CNN-based method to extract the features. Subsequently, using the BPSO_LR proposed algorithm to select the most informative feature subset, and finally, training the classifier using the selected training set and using the same classifier to discriminate between the cover and stego image for the testing set as new unpredicted images.

4.2 System Requirements

The hardware and software requirements to apply the proposed image steganalysis method are illustrated as follows:

- Hardware: Processor Intel ® Core ™ i7-4810MQ, RAM 32 GB, Freq. 2.80 GHz.
- Operating System: Windows 7 (64-bit).
- Programming Language: Python version 3.7 (64-bit).
- IDE: Pycharm 2020.2.3 (Community Edition).

To build the models and extract the features based on the proposed CNN, all the experiments were performed on NVIDIA's Tesla K80 GPU platform to speed up the running and because of the memory limitations.

4.3 Datasets Preparation

Five steganographic methods are used to evaluate the proposed method for image steganalysis. These steganographic methods are Hill, HUGO, MiPOD, S-UNIWARD, and WOW, which were prepared using the BOSSbase v1.01 dataset and considered as content-adaptive methods for the embedding in the spatial domain. BOSSbase dataset contains 10,000 uncompressed natural images, including human beings, natural scenes, animals, and buildings.

For each steganographic algorithm, whether it is a cover or stego, the image dataset is splitted into 70% training and 30% for testing. Table (4.1) shows the datasets splitting for training and testing collections.

Table (4.1): The Dataset Splitting

Set	Cover	Stego	Total
Training Set	7000	7000	14000
Testing Set	3000	3000	6000

4.4 NRF Experimental Results

As mentioned in section (3.2.2), adding NRF as a pre-processing step before applying the CNN architecture can led to a better detection performance. Without using this step, a large number of epochs are needed for convergence of the proposed CNN. To demonstrates the effect of applying NRF on the accuracy and convergence of the network after applying the algorithm (3.3), Figure (4.1) illustrates the learning curves of the CNN experiments in the training process applied by using the WOW

steganographic method with 100 epochs at 10%, 20%, 30%, and 40% bpp embedding rates, respectively.

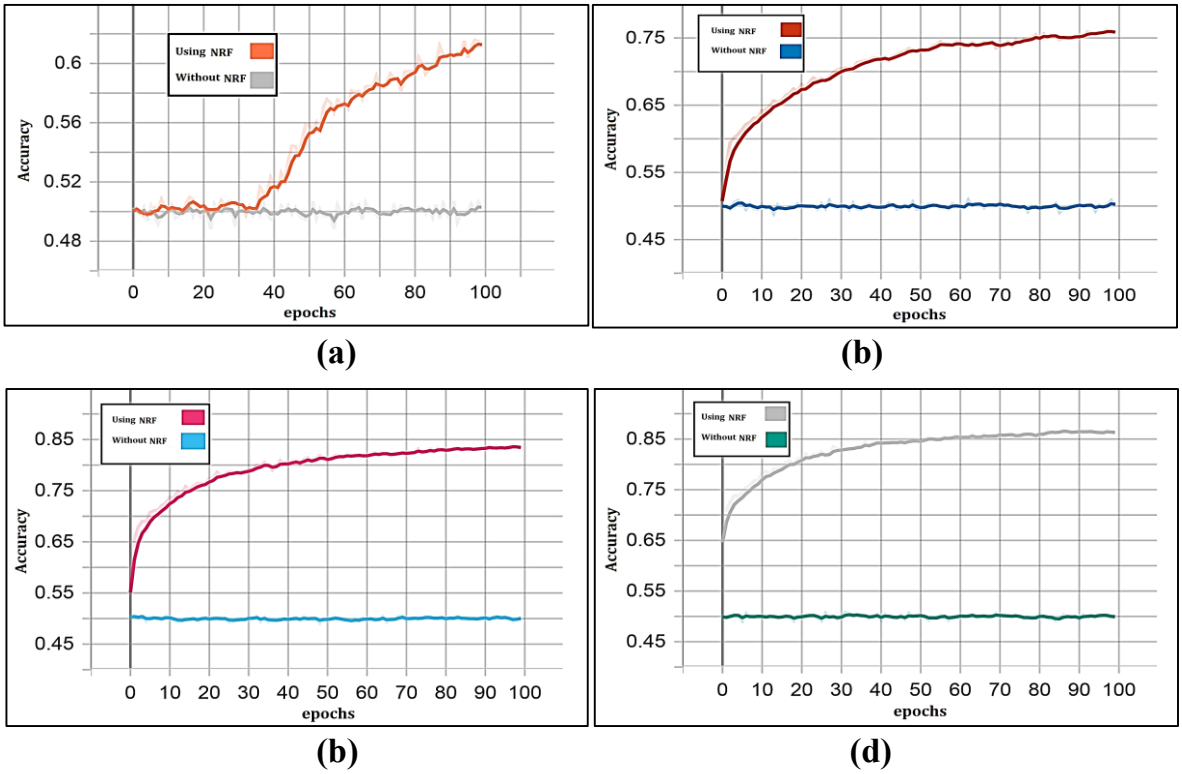


Figure (4.1): The Effect of Applying NRF on Learning the CNN using WOW Steganographic Method on (a)10%, (b) 20%, (c) 30%, and (d) 40% Embedding Rates.

From the curves of the experiments illustrated in Figure (4.1), it is observed that the NRs learning performs better than the case without applying NRF. As a result, this preliminary step is important to the steganalysis problem, and the CNNs converge will be much slower without using this step. Also, Figures (4.2) and (4.3) illustrate the samples of some BOSSbase images before and after applying a NRF on the sample images, respectively.

Figure (4.2): Samples from Bossbase images dataset

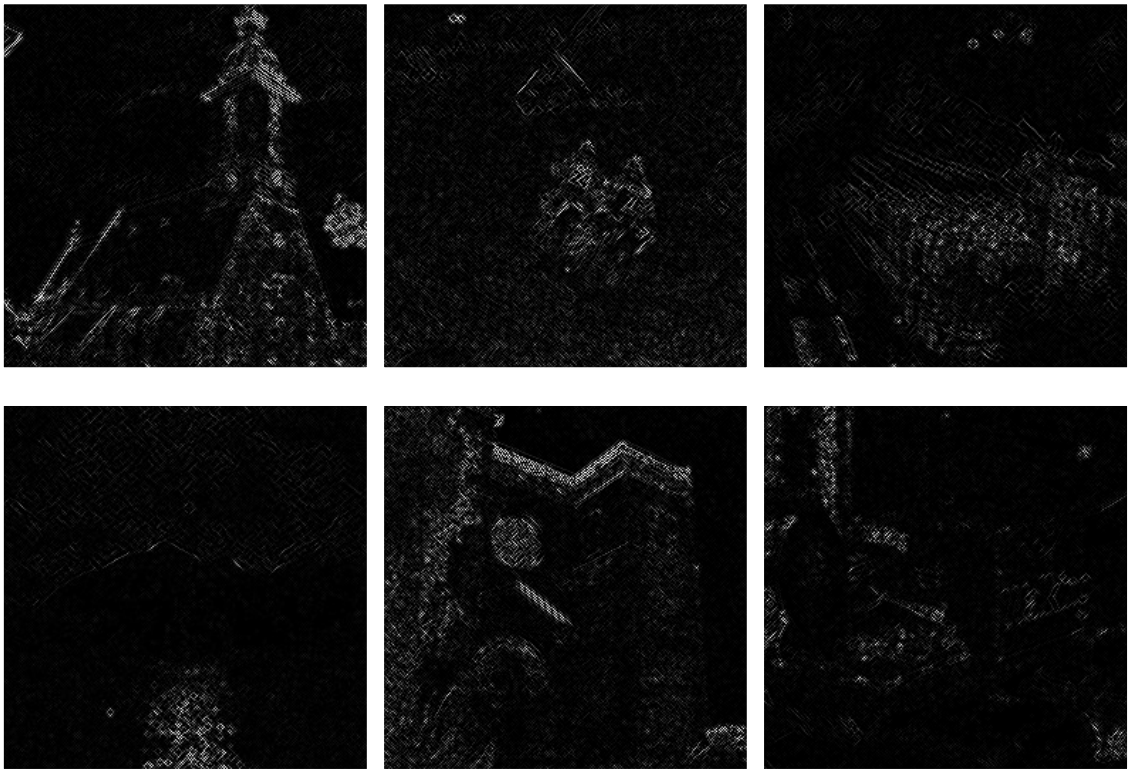


Figure (4.3): Corresponding Images Resulting After Applying NRF on Images in Figure (4.2)

4.5 CNN Experiments

This section aims to evaluate the efficiency of the proposed CNN method to extract the features for image steganalysis through various experiments to illustrate its effectiveness. The dataset and setting the parameters of learning the model are described, and then the proposed method is compared with some previous steganalysis methods based on CNN.

4.5.1 Parameters Setting of the proposed CNN

The training and validation of the proposed CNN model is performed by using Adam optimizer with a learning rate= 0.001, decay= 0.000004, beta_1 = 0.9, beta_2 = 0.999, the epsilon is 1e – 08 and the mini-batch size of 128 for CNN (64 cover/stego pairs). CNN gives up to 250 epochs of preparation. The best training model has been chosen based on the validation range. The datasets are splitted into training sets (70%) and testing sets (30%). Also, 20% from the each training set is used to validate the CNN model.

4.5.2 CNN Experimental Results

The features that discriminate between the "cover" and "stego" images are extracted by applying the experiments on the prepared datasets related to five content-adaptive image steganographic methods after pre-processing with an NRF and normalization. Each experiment is applied using a specified embedding rate, where each of these steganographic methods includes four different embedding rates.

The proposed CNN model is designed to learn appropriate features and use it as a feature extraction (Figure 3.4). The proposed CNN are

constructed from several blocks, and each block contains many layers with different purposes. Table (4.2) demonstrates an overview of these layers based on each module and each included block.

Table (4.2): Overview of the Proposed CNN Layers.

Layer	**Output size**	**Layer Specific Parameters**
Convolutional Module		
Base Block_1 (Two Subnets)		
input_1 (Input Layer)	None, 256, 256, 1	0
conv2d_1	None, 256, 256, 10	250
conv2d_2	None, 256, 256, 10	10
absolute_1	None, 256, 256, 10	0
absolute _2	None, 256, 256, 10	0
activation_1	None, 256, 256, 10	0
activation_2	None, 256, 256, 10	0
average_pooling2d_1	None, 126, 126, 10	0
average_pooling2d_2	None, 126, 126, 10	0
concatenate_1	None, 126, 126, 20	0
Base Block_2		
batch_normalization_1	None, 126, 126, 20	80
conv2d_3	None, 126, 126, 20	10020
activation_3	None, 126, 126, 20	0
average_pooling2d_3	None, 61, 61, 20	0
Base Block_3		
batch_normalization_2	None, 61, 61, 20	80
conv2d_4	None, 61, 61, 30	15030
leaky_re_lu_1	None, 61, 61, 30	0
average_pooling2d_4	None, 29, 29, 30	0

To be continue

Base Block_4 (Two Subnets)		
batch_normalization_3	None, 29, 29, 30	120
conv2d_5	None, 29, 29, 30	22530
conv2d_6	None, 29, 29, 30	930
leaky_re_lu_2	None, 29, 29, 30	0
leaky_re_lu_3	None, 29, 29, 30	0
average_pooling2d_5	None, 13, 13, 30	0
average_pooling2d_6	None, 13, 13, 30	0
concatenate_2	None, 13, 13, 60	0
Base Block_5		
batch_normalization_4	None, 13, 13, 60	240
conv2d_7	None, 13, 13, 64	34624
leaky_re_lu_4	None, 13, 13, 64	0
average_pooling2d_7	None, 5, 5, 64	0
Base Block_6		
batch_normalization_5	None, 5, 5, 64	256
conv2d_8	None, 5, 5, 128	8320
leaky_re_lu_5	None, 5, 5, 128	0
average_pooling2d_8	None, 1, 1, 128	0
Base Block_7		
batch_normalization_6	None, 1, 1, 128	512
conv2d_9	None, 1, 1, 256	33024
leaky_re_lu_6	None, 1, 1, 256	0
global_average_pooling2d_1	None, 256	0
Classification Module		
dense_1	None, 256	65792
dropout_1	None, 256	0
leaky_re_lu_7	None, 256	0
dense_2	None, 256	65792
dropout_2	None, 256	0
leaky_re_lu_8	None, 256	0
dense_3	None, 2	514
Softmax (Output Layer)	None, 2	0

The filtered image is convolved with the Convolution layer to produce a feature map. Also, the non-linearity has been introduced by using the non-linear activation function in each base block. The pooling operation is applied on the learned feature map using the AVG-pooling, which is mostly used in steganalysis models to obtain the steganographic noise of the images or use the GAVG-pooling layer to down-sampling the feature map. Two subnets that are repeatedly combined and separated are used to introduce the diversity of the extracted features as mentioned in the algorithm (3.4). The classification module, which is comprised of two fully connected and a softmax layers, converts feature vectors to posterior probabilities for each class (algorithm 3.5). The final class labels are obtained by selecting the class with the highest posterior. The Accuracy is computed for both training and validation sets and then for testing sets to analyze the performance of the proposed CNN method as a feature extraction stage for steganalysis of content-adaptive image steganography.

There are many experiments conducted to extract the features by using the proposed CNN. The accuracy curves for the training and validation sets of CNN are illustrated in Figures (4.4), (4.5), (4.6), (4.7), and (4.8) that represent applying HILL, HUGO, MiPOD, S-UNIWARD, and WOW steganographic methods respectively. The x-axis represents the number of epochs, and the y-axis represents the detection accuracy.

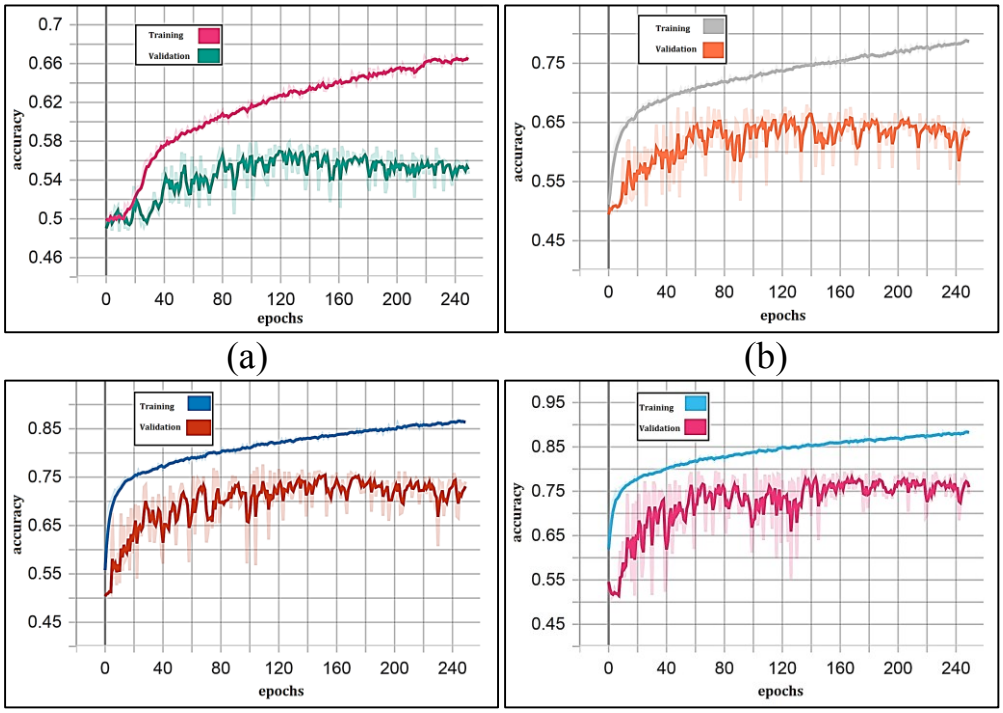


Figure (4.4): The Training and Validation Accuracy values of HILL Stego Algorithm at 10%, 20%, 30%, and 40% Embedding Rates

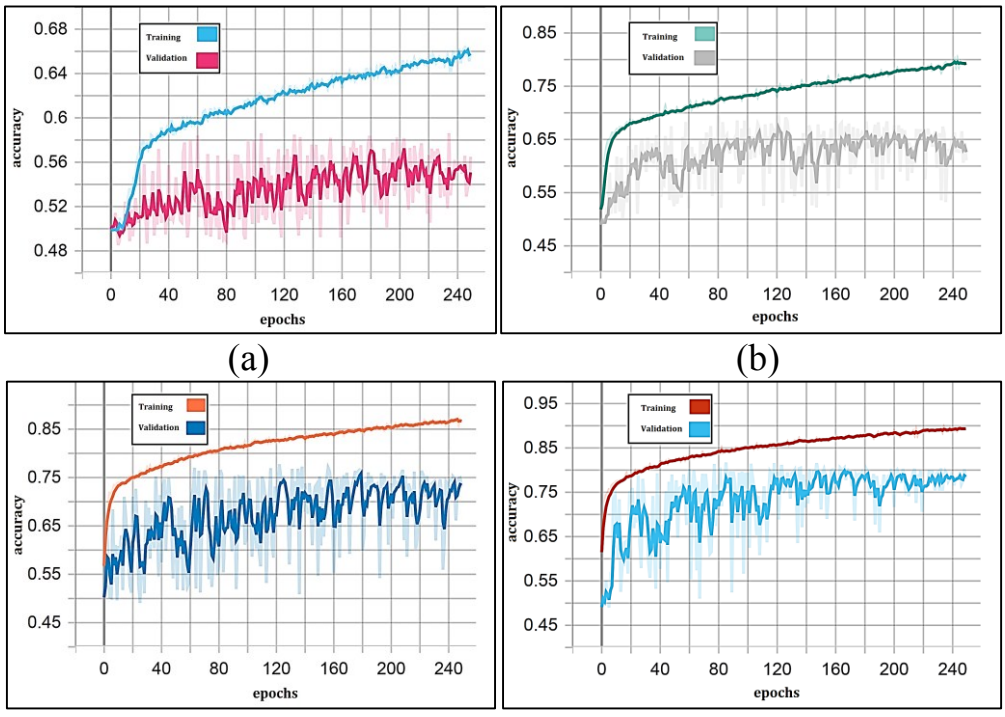


Figure (4.5): The Training and Validation Accuracy Values of HUGO Stego Algorithm at 10%, 20%, 30%, and 40% Embedding Rates

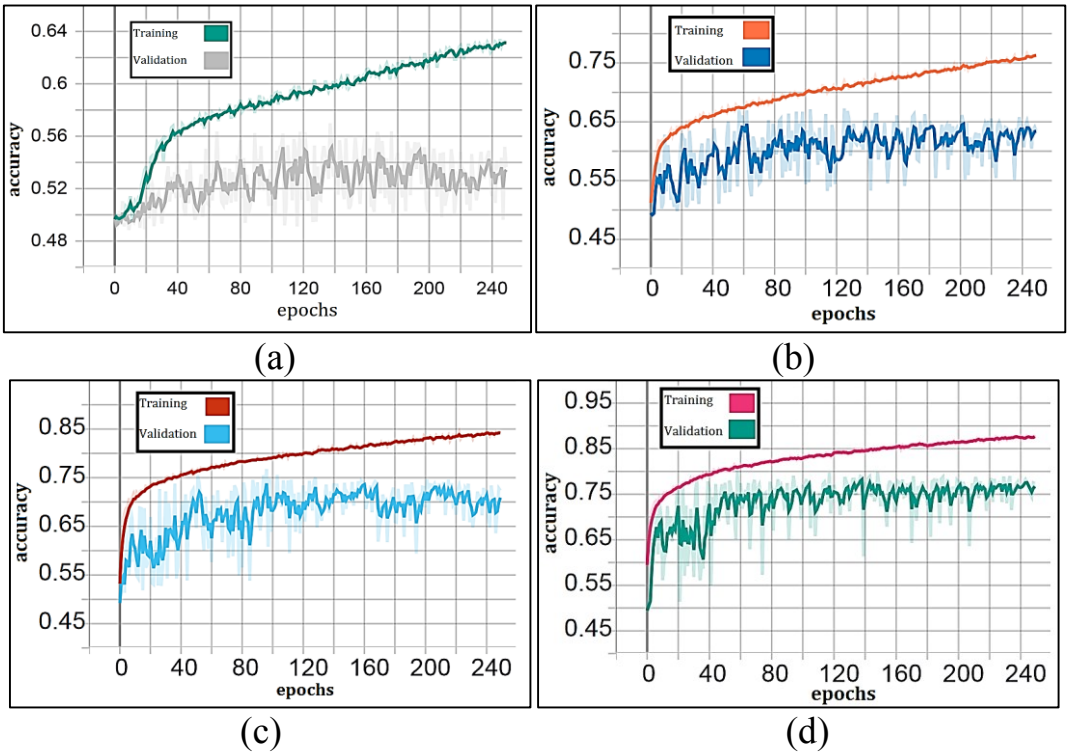


Figure (4.6): The Training and Validation Accuracy Values of MiPOD Stego Algorithm at 10%, 20%, 30%, and 40% Embedding Rates

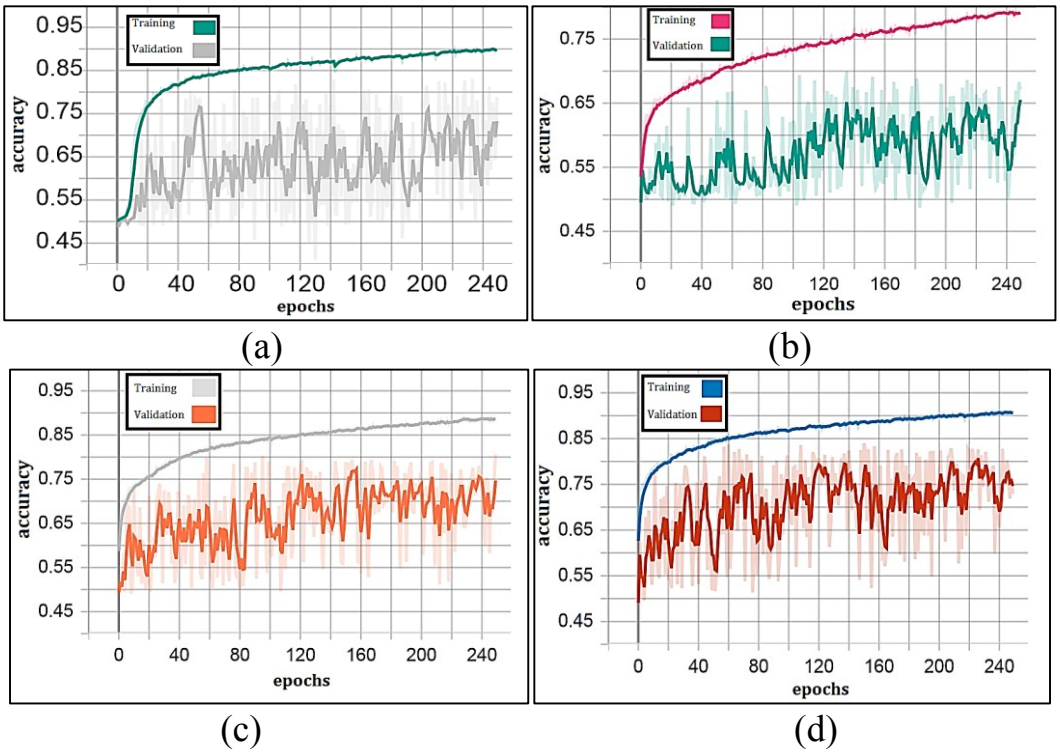


Figure (4.7): The Training and Validation Accuracy Values of S-UNIWARD Stego Algorithm at 10%, 20%, 30%, and 40% Embedding Rates

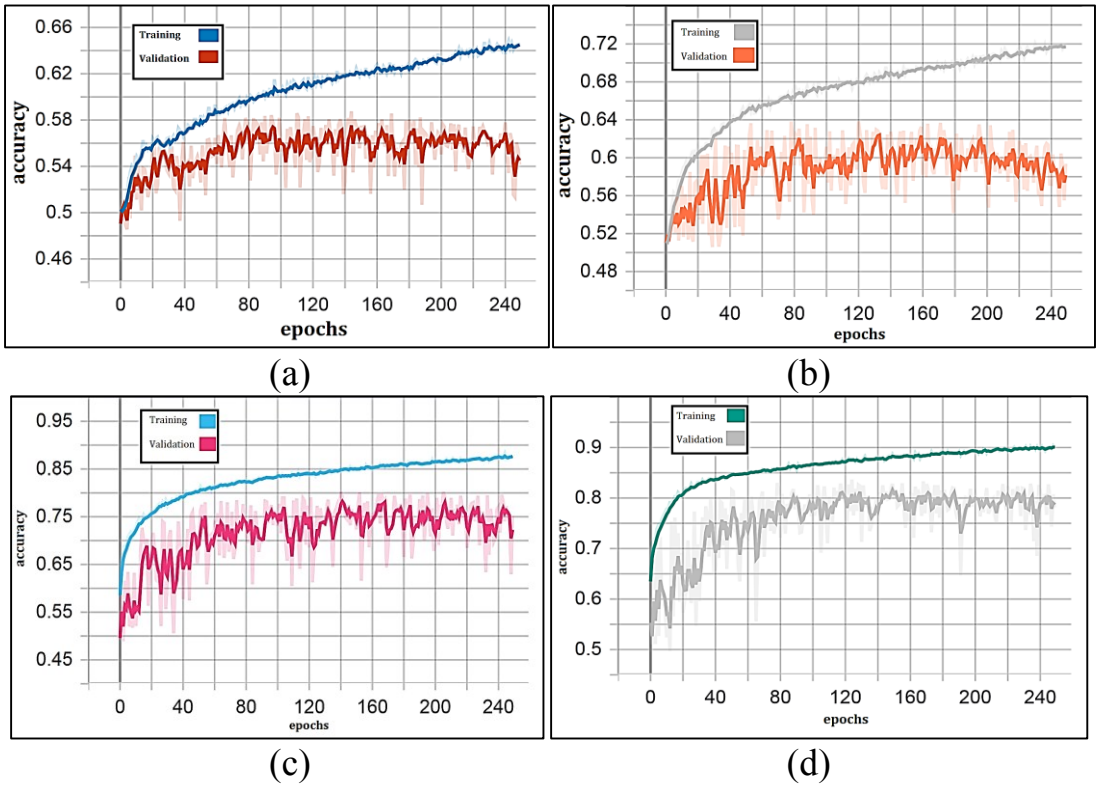
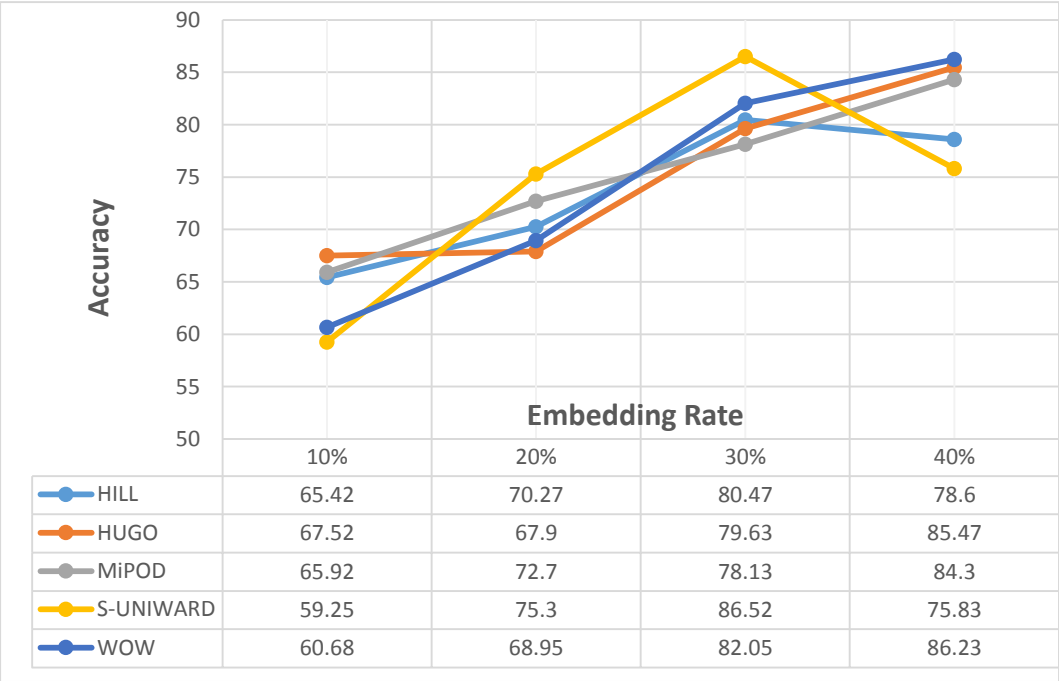


Figure (4.8): The Training and Validation Accuracy Values of WOW Stego Algorithm at 10%, 20%, 30%, and 40% Embedding Rates

The learning process is performed as illustrated in algorithm (3.4). As shown in the curves of these Figures, the gap between the training and validation is increased slightly at low embedding rates, especially for 10% embedding rate, as the number of epochs increases, and this gap goes down as the embedding rate increases. This problem occurs because CNN becomes harder to learn features to detect the difference between stego and cover images at low embedding rates.

To evaluate the performance of the proposed CNN, a testing set is predicted after learning the CNN model. Figure (4.9) records the accuracy results by using the datasets of HILL, HUGO, MiPOD, S-UNIWARD, and WOW content-adaptive steganographic methods at 10%, 20%, 30%, and 40% bpp embedding rates.

It is common knowledge that when the embedding rate increases, the detection accuracy also increases. From Figure (4.9), it is observed that the detection accuracy decreases by using HILL and S-UNIWARD methods at 40%, and this occurs due to the presence of redundant and irrelevant features that overwhelm the performance of the useful information provided by informative features and hence reduces the quality of the whole feature set. As what will be illustrated in the next sections, a FS strategy is used to overcome this problem, which has proven its effectiveness.



	10%	20%	30%	40%
HILL	65.42	70.27	80.47	78.6
HUGO	67.52	67.9	79.63	85.47
MiPOD	65.92	72.7	78.13	84.3
S-UNIWARD	59.25	75.3	86.52	75.83
WOW	60.68	68.95	82.05	86.23

Figure (4.9): Accuracy Values using the CNN proposed method Resulted from Testing Five Stego Algorithms at Four Embedding Rates

Moreover, the performance is compared with other CNN-based image steganalysis methods for further evaluate of the proposed method. Table (4.3) illustrates the detection accuracy comparisons using HILL, HUGO, MiPOD, S-UNIWARD, and WOW steganographic methods at

10%, 20%, and 40% embedding rates. The bolded results refer to the accuracy values that satisfy the best performance among the corresponding compared methods.

Table (4.3): Comparison Accuracy Values on Different Stego Algorithms at 10%, 20%, 30% and 40% Embedding Rates

CNN-based Steganalysis Method	**Stego Algorithm**	**Embedding Rates**			
		10%	**20%**	**30%**	**40%**
Method in [15]	HILL	58.44	-	-	**79.24**
	S-UNIWARD	57.33	-	-	**80.24**
The Method in [24]	MiPOD	-	62.3	68.9	75.1
	S-UNIWARD	-	62.1	69.8	73.3
	WOW	-	61.2	66.3	72.6
IAS-CNN [25]	HUGO	-	**70.45**	-	78.35
	S-UNIWARD	-	62.4	-	75.05
	WOW	-	68.15	-	80.75
DFSE_Net [27]	S-UNIWARD	57.8	65.9	71.6	78.5
	WOW	**65.8**	**75.3**	80.7	85.1
Proposed Method	HILL	**65.42**	-	-	78.6
	HUGO	-	67.9	-	**85.47**
	MiPOD	-	**72.7**	**78.13**	**84.3**
	S-UNIWARD	**59.25**	**75.3**	**86.52**	75.83
	WOW	60.68	68.95	**82.05**	**86.23**

As shown in Table (4.3), the proposed CNN method generally satisfies better performance, at most embedding rates, when compared to other CNN-based steganalysis methods. Also, some other results have different performance ranging from less than, and close to the other

compared methods. These different performance levels are due to, as previously mentioned, the presence of redundant and irrelevant features.

4.6 Feature Selection Experiments

Because of hyperparameters setting, the extracted features by CNN may not be general enough to fit the steganalysis for most steganographic methods to the provided results that are always perfect. As observed in the CNN experiments, the proposed method given an improvement compared with some other CNN-based steganalysis methods. However, there is a decrease in the level of performance in some embedding rates, especially for 40% embedding rate, after applying the proposed CNN method to some steganographic methods. This degraded performance is due to the presence of redundant and irrelevant extracted features. Therefore, to solve this issue, the extracted features are fed to a FS stage to eliminate these redundant and irrelevant features and enhance steganalysis performance.

The proposed method for FS is based on uses BPSO with LR classifier (BPSO_LR) as a fitness function. The input to this algorithm includes a training set resulting from applying CNN, has 256-D best extracted features, and the output consists of the best feature subset for steganalysis purposes. Therefore, the experiments to illustrate the efficiency of the proposed BPSO_LR algorithm have been presented and analyzed. The value of fitness function is calculated using LR classifier based on 10-fold cross-validation. However, for all experiments, the population size and number of iterations are set to 15, and 30, respectively. Each experiment has repeated five times, and the highest obtained accuracy value was selected from these experiments to be the final result.

As this algorithm is stochastic in nature, average fitness (mean) and standard deviation (STD) are computed for each executed fitness function to illustrate the performance of the proposed algorithm for FS. The algorithm's robustness and consistency are represented by the STD. As a consequence, the FS algorithm with the lowest STD value has more robustness and provides the most consistent results. On the one hand, a greater accuracy score indicates that more samples have been predicted accurately. Table (4.4) illustrates some samples of mean and STD resulting in the outputs from applying LR as a fitness function using 10-fold cross-validation after applying BPSO_LR on the MiPOD steganographic algorithm at 20% embedding rate.

In terms of mean and STD fitness values, BPSO_LR has a competitive performance, as shown in Table (4.4). The results show that BPSO_LR satisfy an optimal performance in selecting relevant features, resulting in promising results.

A meta-heuristic algorithm's convergence behavior is also a critical issue that has to be demonstrated. As a result, the proposed BPSO_LR has also been examined by plotting the convergence graph for each steganographic algorithm on each embedding rate. Figures (4.10), (4.11), (4.12), (4.13), and (4.14) demonstrate the convergence trends for each steganographic algorithm on each embedding rate for 30 iterations. The best fitness values acquired till the current generation are shown on the y-axes of the converging curves. From the convergence curves, it can be noticed that by applying the proposed BPSO_LR algorithm on the different methods, most of them converge after few iterations.

Table (4.4): Some 10-fold cross-validation Samples Resulting from Applying LR within BPSO on MiPOD Steganographic Algorithms at 20% Embedding Rate.

10-folds	Mean	STD
0.6871428571428572 0.7114285714285714 0.6892857142857143 0.6871428571428572 0.6757142857142857 0.6757142857142857 0.69 0.6614285714285715 0.6707142857142857 0.6757142857142857	0.682429	(+/- 0.01)
0.6778571428571428 0.7042857142857143 0.6907142857142857 0.6642857142857143 0.6792857142857143 0.6707142857142857 0.6835714285714286 0.6692857142857143 0.6757142857142857 0.6692857142857143	0.678500	(+/- 0.01)
0.6771428571428572 0.6985714285714286 0.6828571428571428 0.6592857142857143 0.6742857142857143 0.6721428571428572 0.6821428571428572 0.6764285714285714 0.6857142857142857 0.675	0.678357	(+/- 0.01)
0.6871428571428572 0.7035714285714286 0.6878571428571428 0.655 0.67 0.6528571428571428 0.6978571428571428 0.675 0.67 0.6628571428571428	0.676214	(+/- 0.02)
0.6857142857142857 0.6985714285714286 0.6807142857142857 0.6707142857142857 0.6835714285714286 0.6785714285714286 0.6921428571428572 0.6685714285714286 0.6764285714285714 0.6742857142857143	0.680929	(+/- 0.01)
0.685 0.7014285714285714 0.6928571428571428 0.6685714285714286 0.6664285714285715 0.6735714285714286 0.6814285714285714 0.6564285714285715 0.6664285714285715 0.675	0.676714	(+/- 0.01)
0.6792857142857143 0.6978571428571428 0.6778571428571428 0.6671428571428571 0.6735714285714286 0.6807142857142857 0.6885714285714286 0.6692857142857143 0.6735714285714286 0.6742857142857143	0.678214	(+/- 0.01)

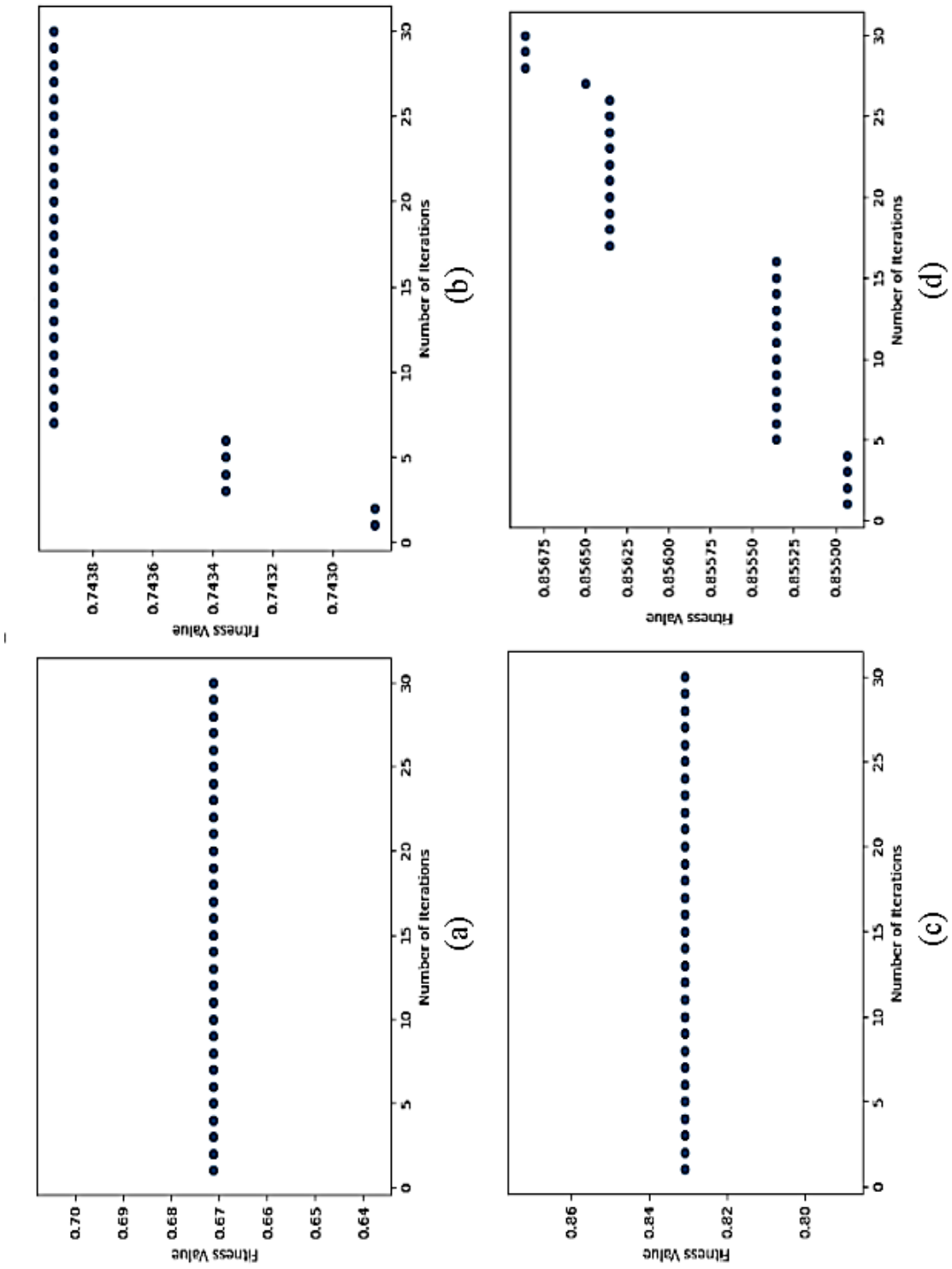


Figure (4.10): The Convergence Curves for Applying BPSO_LR at (a) 10% (b) 20% (c) 30% and (d) 40% Embedding Rates respectively on HILL Stego Algorithm.

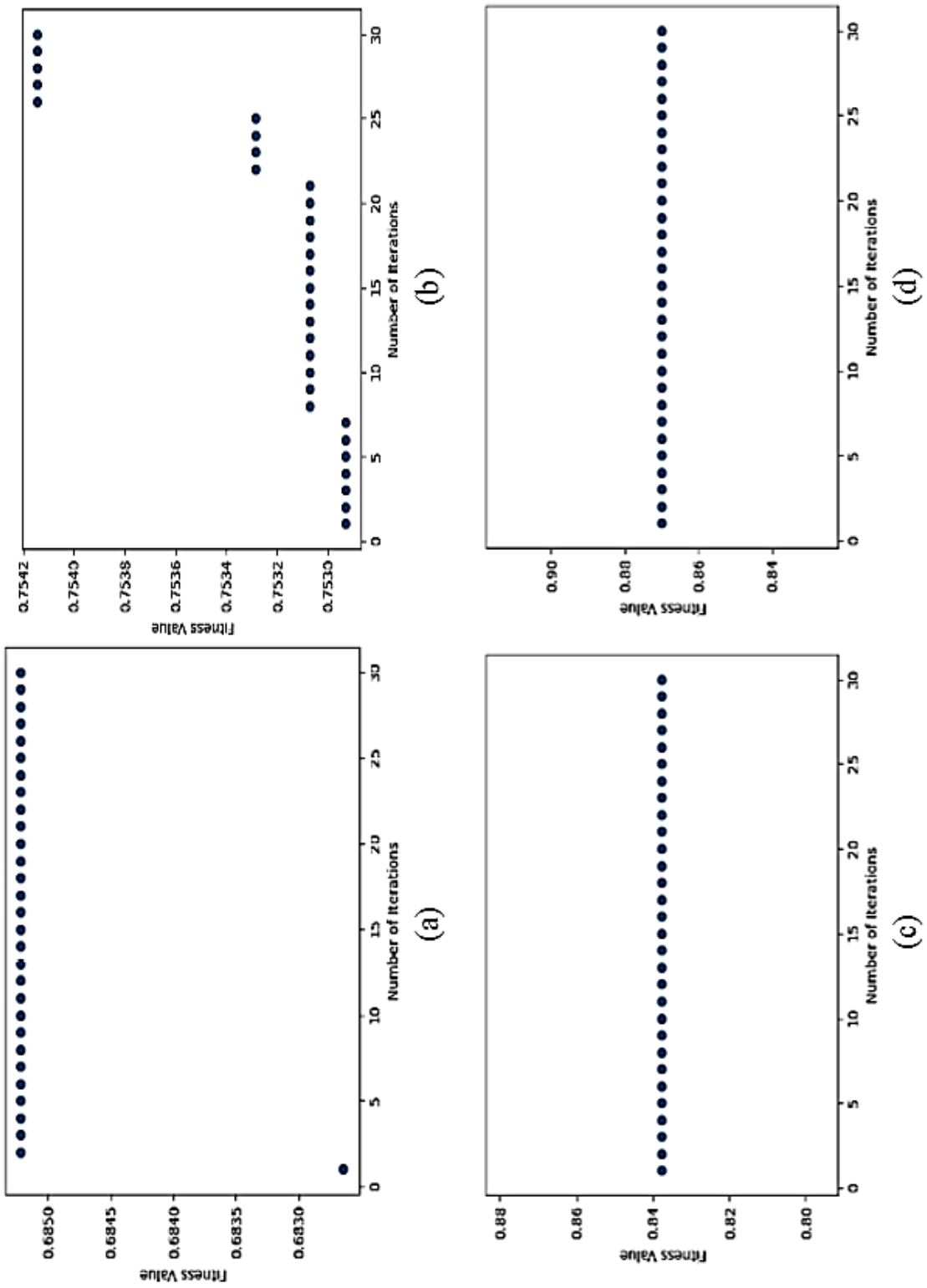


Figure (4.11): The Convergence Curves for applying BPSO_LR at (a)10% (b)20% (c)30% and (d)40% Embedding Rates respectively on HUGO Stego Algorithm.

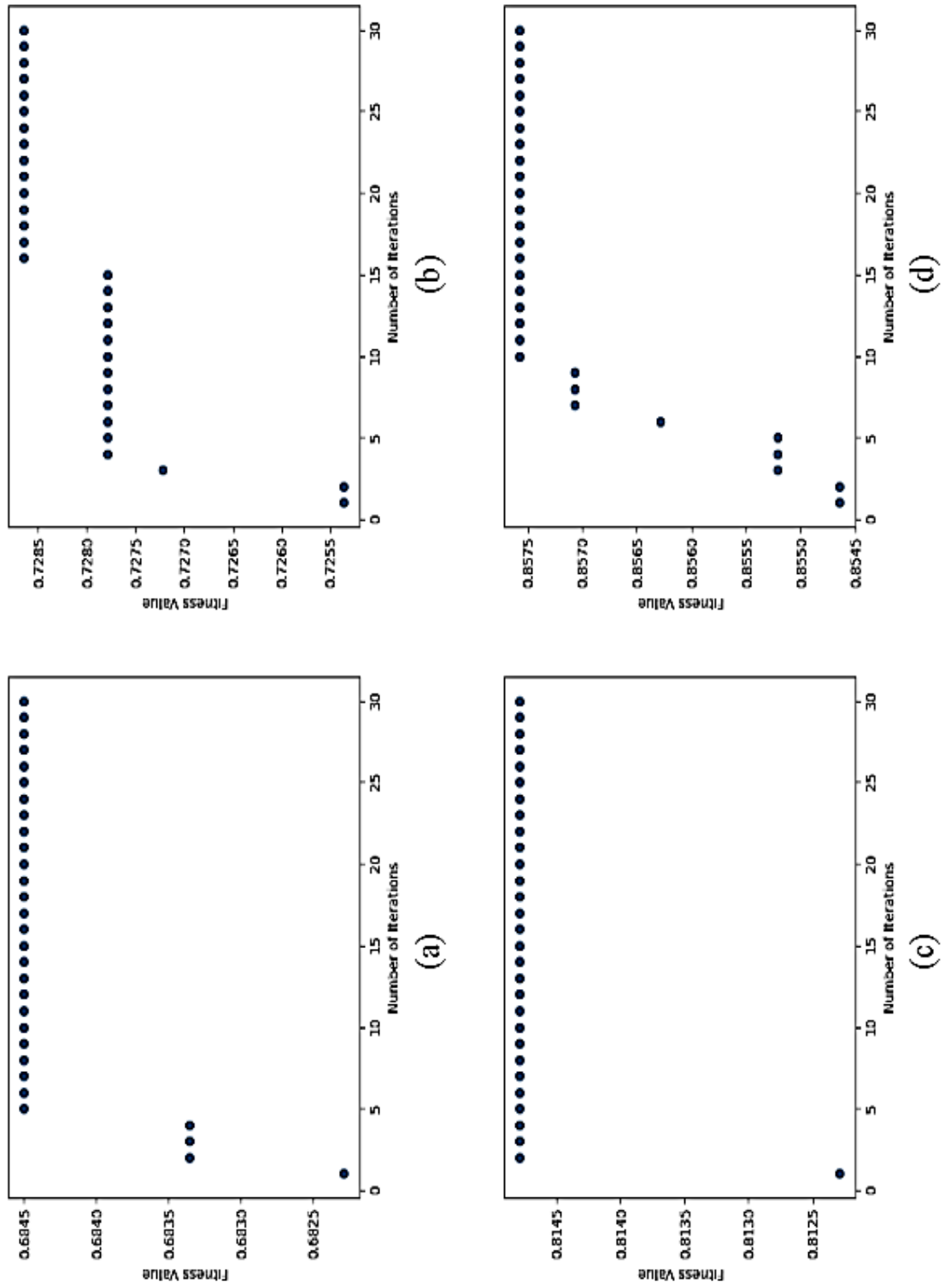


Figure (4.12): The Convergence Curves for applying BPSO_LR at (a)10% (b)20% (c)30% and (d)40% Embedding Rates on MiPOD Stego Algorithm.

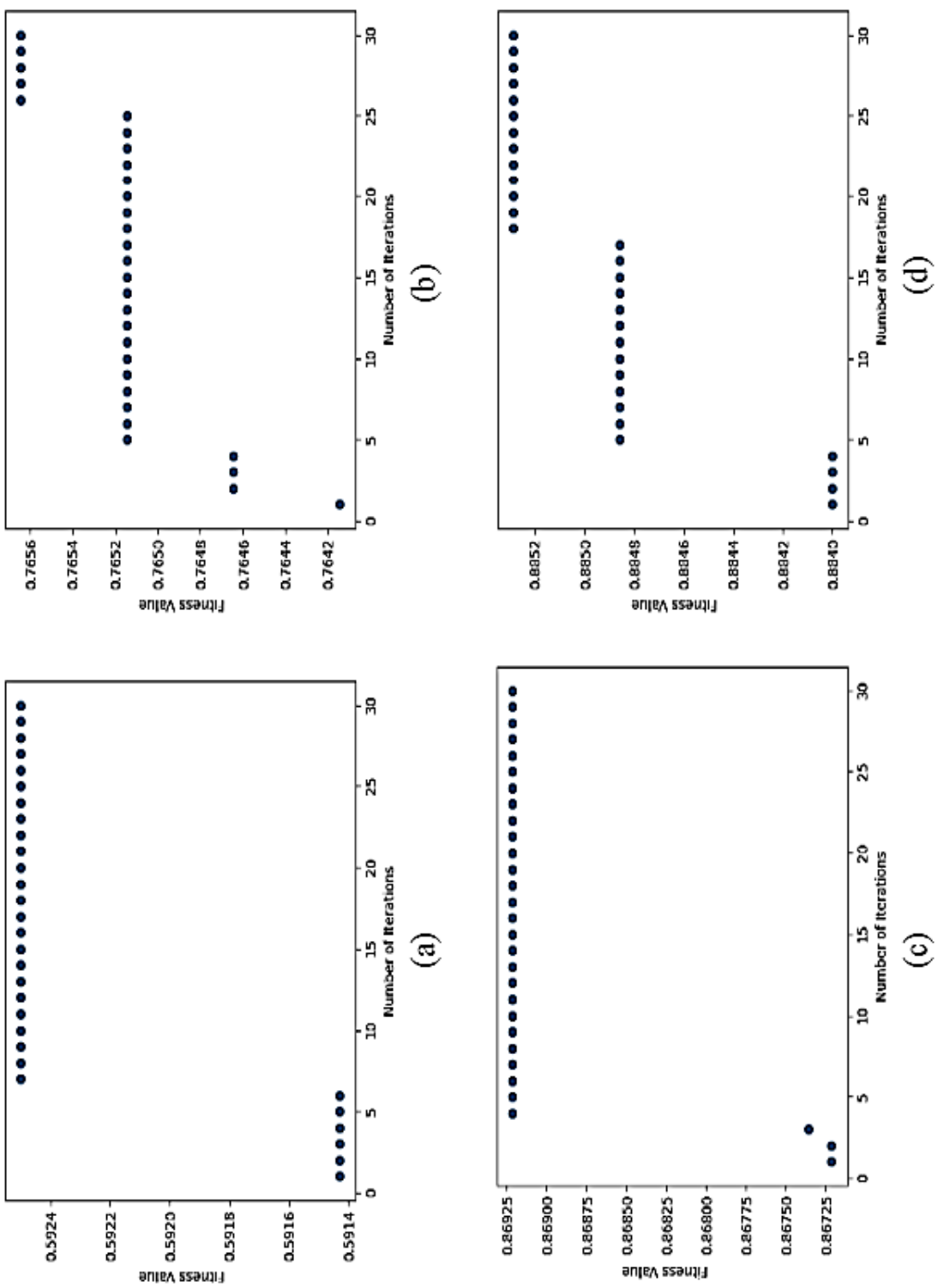


Figure (4.13): The Convergence Curves for applying BPSO_LR at (a)10% (b)20% (c)30%, and (d)40% Embedding Rates, respectively, on S-UNIWARD Stego Algorithm.

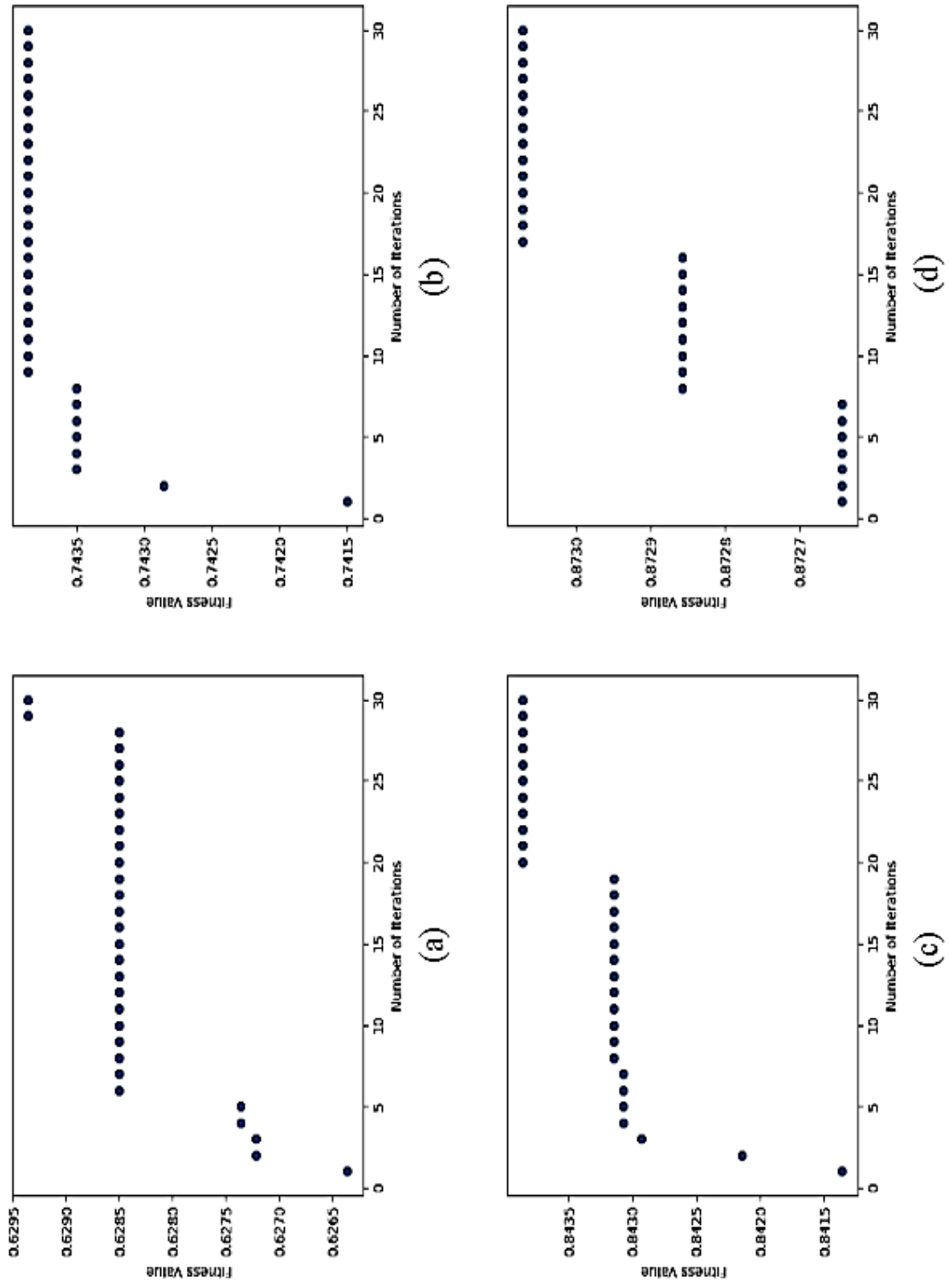


Figure (4.14): The Convergence Curves for applying BPSO_LR at: (a) 10% (b) 20% (c) 30%, and (d) 40% Embedding Rates, respectively, on WOW Steganographic Algorithm.

The performance of BPSO_LR FS algorithm has been demonstrated by the number of selected features and the iteration through which they are converged. From the results that are shown in Table (4.5), it can be seen that the proposed FS algorithm contributed to reduce the number of features between 40.63% to 59.38%, where the number of selected features ranges between 104 and 152 features.

Table (4.5): FS and the Convergence Resulting from Applying BPSO_LR on each Stego Algorithm and Embedding Rate.

Steganographic Algorithm	**Embedding Rate**	**No. of Selected Features out of 256**	**Convergence Iteration**
HILL	10%	124	1
	20%	145	7
	30%	131	1
	40%	124	28
HUGO	10%	135	2
	20%	152	26
	30%	127	1
	40%	129	1
MiPOD	10%	104	5
	20%	111	16
	30%	128	2
	40%	128	10
S-UNIWARD	10%	128	7
	20%	121	26
	30%	142	4
	40%	127	18
WOW	10%	128	29
	20%	133	9
	30%	133	20
	40%	125	17

4.7 Experiments of Classification Results

Many experiments have been performed using the SVM classifier to demonstrate the efficiency of the proposed method for FS. The choosing of the SVM classifier is due to it achieves higher results than other machine learning classifiers and the final classification has been performed according to the algorithm (3.6). Table (4.6) shows the detection accuracy before and after applying the features selection using the proposed BPSO_LR applied on HILL, HUGO, MiPOD, S-UNIWARD, and WOW content-adaptive steganographic algorithms at four embedding rates. Comparing with the results that are illustrated in Figure (4.9), it has been noticed that there is an improvement in performance when applying BPSO_LR for FS. The improvement ranging from 3.46% to 7.36%, from 2.06% to 8.72%, from 0.82% to 5.3%, from 0.64% to 12.33%, and from 0.81% to 7.11% for HILL, HUGO, MiPOD, S-UNIWARD, and WOW steganographic algorithms, respectively. This variety in performance improvement is due to the variety in the number of redundant and irrelevant features found on each steganographic method and each embedding rate. Also, it is noticed that the degradation in the detection accuracy after applying CNN that occurred in some embedding rates, such as accuracy resulted from applying 40% embedding rate using HILL and S-UNIWARD steganographic methods, it has been processed at this stage, where there was a significant increase in detection accuracy for those were suffering from this degradation.

To justify the accuracy of applying the SVM classifier, other experiments have been conducted using NB, and RF classifiers to calculate the classification accuracy, as illustrated in Table (4.7). For the

features selected by the proposed algorithm, the accuracy obtained by the SVM classifier is better than the accuracy obtained by the others.

Table (4.6): Accuracy of the Proposed Method before and after Applying the proposed BPSO_LR on CNN features for each Stego Algorithm

Steganographic Method	Embedding Rate	Detection Accuracy		Improving Rate %
		CNN	CNN+BPSO_LR+SVM	
HILL	10%	65.42	69.80	4.38
	20%	70.27	76.45	6.18
	30%	80.47	83.93	3.46
	40%	78.6	85.96	7.36
HUGO	10%	67.52	71.00	3.48
	20%	67.9	76.62	8.72
	30%	79.63	84.13	4.5
	40%	85.47	87.53	2.06
MiPOD	10%	65.92	71.22	5.3
	20%	72.7	74.94	2.24
	30%	78.13	82.61	4.48
	40%	84.3	85.12	0.82
S-UNIWARD	10%	59.25	60.77	1.52
	20%	75.3	77.49	2.19
	30%	86.52	87.16	0.64
	40%	75.83	88.16	12.33
WOW	10%	60.68	65.39	4.71
	20%	68.95	76.06	7.11
	30%	82.05	83.85	1.8
	40%	86.23	87.04	0.81

Table (4.7): Comparison of the Accuracy using Different Classifiers after Applying BPSO_LR on CNN features for each Stego Algorithm

Steganographic Algorithm	Embedding Rate	Detection Accuracy		
		NB	RF	SVM
HILL	10%	64.85	50.28	**69.80**
	20%	71.25	62.00	**76.45**
	30%	77.85	74.93	**83.93**
	40%	80.78	78.28	**85.96**
HUGO	10%	65.28	53.18	**71.00**
	20%	69.55	62.9	**76.62**
	30%	79.55	75.5	**84.13**
	40%	82.33	80.62	**87.53**
MiPOD	10%	66.75	53.48	**71.22**
	20%	71.0	58.92	**74.94**
	30%	76.72	72.18	**82.61**
	40%	80.17	78.05	**85.12**
S-UNIWARD	10%	58.57	38.4	**60.77**
	20%	70.47	65.47	**77.49**
	30%	84.87	80.92	**87.16**
	40%	80.3	82.83	**88.16**
WOW	10%	61.63	44.38	**65.39**
	20%	68.43	63.02	**76.06**
	30%	79.87	77.53	**83.85**
	40%	85.2	81.57	**87.04**

The Recall, Precision, and F-measures are computed based on confusion matrix contents. These metrics are illustrated in equations 2.42 , 2.43, and 2.44, respectively, which considered another evaluation of the proposed system. Table (4.8) illustrates these metrics values using SVM for five stego algorithms and fort each embedding rate, and the time cost required for testing.

Table (4.8): Precision, Recall, F-Measure, and time cost using SVM Classifier after Applying BPSO_LR on CNN features for each Stego Algorithm.

Steganographic Algorithm	Embedding Rate	Detection Accuracy using SVM			Time/ Second
		Precision%	Recall%	F-Measure%	
HILL	10%	62.47	79.07	69.80	37.195
	20%	70.73	83.19	76.45	34.505
	30%	78.82	89.76	83.93	21.591
	40%	81.68	90.71	85.96	17.187
HUGO	10%	64.63	78.77	71.00	38.640
	20%	70.02	84.62	76.62	36.347
	30%	79.54	89.28	84.13	19.798
	40%	82.26	93.54	87.53	15.087
MiPOD	10%	63.01	81.9	71.22	28.585
	20%	68.71	82.41	74.94	28.61
	30%	76.9	89.25	82.61	22.835
	40%	82.04	88.43	85.12	17.432
S-UNIWARD	10%	60.37	57.94	60.77	45.316
	20%	73.23	82.27	77.49	27.347
	30%	83.27	91.43	87.16	17.418
	40%	85.51	90.98	88.16	11.921
WOW	10%	60.79	70.74	65.39	39.139
	20%	72.14	80.44	76.06	30.707
	30%	81.00	86.9	83.85	18.484
	40%	85.25	88.91	87.04	13.54

4.8 Comparison with the Related Works

In this section, a comparison between the generated results of the proposed method and other related works for steganalysis over the steganographic datasets constructed based on the BOSSbase dataset is presented through many experiments.

The comparisons have been presented in Table (4.9) to illustrate the proposed method's performance compared to some other methods that use feature subset selection for image steganalysis, which uses methods other than CNN to extract the features. From this Table, it has been observed that there are apparent differences in accuracy values in favor of the proposed approach over all other methods, which indicates the importance of this method in improving the performance.

Table (4.9): Accuracy Comparison with FS-based Steganalysis Methods at Different Embedding Rates and Stego Algorithms

Steganalysis Method	**Stego Algorithm**	**Embedding Rates**			
		10%	**20%**	**30%**	**40%**
TLBP [17]	HILL	58.94	66.01	72.09	77.05
	MiPOD	59.25	66.39	72.32	77.29
	S-UNIWARD	59.38	67.88	75.06	81.07
αSRM [20]	HILL	64.67	71.25	76.2	80.27
	MiPOD	59.94	66.69	72.33	77.24
	S-UNIWARD	**63.75**	71.31	76.84	81.34
F_opt [21]	HUGO	63.36	-	78.05	-
	MiPOD	58.08	-	70.21	-
	S-UNIWARD	59.1	-	72.96	-
Method in [22]	HILL	-	-	-	64.11
	HUGO	-	-	-	68.08
	WOW	-	-	-	64.07
Proposed Method	HILL	**69.80**	**76.45**	**83.93**	**85.96**
	HUGO	**71.00**	-	**84.13**	**87.53**
	MiPOD	**71.22**	**74.94**	**82.61**	**85.12**
	S-UNIWARD	60.77	**77.49**	**87.16**	**88.16**
	WOW	**65.39**	**76.06**	**83.85**	**87.04**

CHAPTER FIVE

CONCLUSIONS & SUGGESTIONS FOR FUTURE WORK

Chapter Five
Conclusions and Future Works

5.1 Conclusions

The most important characteristics that have been drawn from the experimental results of the proposed method are presented in the following conclusions:

1. The experiments implemented by using NRF have an important role in improving CNN's performance by suppressing the image contents to highlight the noise components in the image and, therefore, increasing the network learning speed. This conclusion is evidenced by the results of the experiments shown in Figure (4.1).

2. It was important to use different kernel sizes applied in the convolution layer to diversify the extracted features. This diversification with the use of FS algorithms contributes to improve the performance. This conclusion was demonstrated by using two subnets within CNN with two kernel sizes and using the proposed BPSO_LR algorithm for FS.

3. The important aspect of extracting good features using a CNN is the design of the CNN architecture. However, different points of view in the design of these architectures will generates some features that are that obscure the information provided by relevant features; therefore, these features still do not meet the expectations of steganalyzers to be general to discover most steganographic methods. As a result, FS methods are the perfect solution to satisfy more consistent results against most steganographic algorithms. This conclusion is demonstrated in table (4.6)

through the superior results in detection accuracy after applying FS using five steganographic methods and at different embedding rates.

4. The experiments have been proven that the combination of deep learning methods (CNN) as a feature extraction along with swarm intelligence methods (BPSO_LR) as FS and traditional machine learning (SVM) as a classifier can be best effective in image steganalysis.

5. Using SVM for the final classification of selected features had achieved the highest result of prediction when compared with NB and RF classifiers. This conclusion is demonstrated in table (4.7) through the superior results of SVM classifier over other compared classifiers.

3.2 Suggestions for Future Work

This dissertation implements a method for steganalysis of content-adaptive image steganography based on deep learning for feature extraction, swarm intelligence algorithm for FS, and a classical machine learning model for classification. There are several suggestions for future works:

1. Employing the stages of the proposed method for steganalysis of video rather than the image. This may be satisfied either by using it directly or by transfer learning, which is a technique that allows reusing a trained model to solve a similar problem.

2. Rather than using CNN model for each steganographic method and each embedding rate, developed one model that combine the characteristics of these many models and using the softmax function for multi-class prediction.

3. Diversify the feature extraction sources to include the transform domain or extract the features by the statistical methods, and then aggregate these features as input to LSTM or RNNs, which used as a prediction purpose.

4. Improve the proposed method by learning model on high embedding rates and reuse it for prediction in lower embedding rates. This method, along with the FS stage, can increase detection accuracy and processing time efficiency.

5. Provide a visual explanation of CNN model predictions includes visualizing the regions that have been changed during the hiding process. These regions the predictions from these proposed models. This techniques known as Gradient-weighted Class Activation Mapping (Grad-CAM).

REFERENCES

REFERENCES

[1] A. Cheddad, J. Condell, K. Curran, and P. J. S. p. Mc Kevitt, "Digital image steganography: Survey and analysis of current methods," vol. 90, no. 3, pp. 727-752, 2010.

[2] W. Zhang, Z. Zhang, L. Zhang, H. Li, N. J. I. T. o. C. Yu, and S. f. V. Technology, "Decomposing joint distortion for adaptive steganography," vol. 27, no. 10, pp. 2274-2280, 2016.

[3] R. Anderson, "Stretching the limits of steganography," in *International Workshop on Information Hiding*, 1996, pp. 39-48: Springer.

[4] R. J. Anderson and F. A. J. I. J. o. s. a. i. c. Petitcolas, "On the limits of steganography," vol. 16, no. 4, pp. 474-481, 1998.

[5] F. Begum and G. R. J. A. o. t. R. S. f. C. B. Suthoju, "Types of Steganography for Secure Data Maintenance," vol. 25, no. 6, pp. 2144-2159, 2021.

[6] H. J. I.-T. Zhao, Dec, "Security in telecommun-ications and information technology," 2003.

[7] M. Dalal, M. J. M. T. Juneja, and Applications, "Steganography and Steganalysis (in digital forensics): a Cybersecurity guide," vol. 80, no. 4, pp. 5723-5771, 2021.

[8] R. Cogranne, V. Sedighi, J. Fridrich, and T. Pevný, "Is ensemble classifier needed for steganalysis in high-dimensional feature spaces?," in *2015 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2015, pp. 1-6: IEEE.

[9] J. Kodovsky, J. Fridrich, V. J. I. T. o. I. F. Holub, and Security, "Ensemble classifiers for steganalysis of digital media," vol. 7, no. 2, pp. 432-444, 2011.

[10] M. Chaumont, "Deep learning in steganography and steganalysis," in *Digital Media Steganography*: Elsevier, 2020, pp. 321-349.

[11] V. Holub, J. Fridrich, and T. J. E. J. o. I. S. Denemark, "Universal distortion function for steganography in an arbitrary domain," vol. 2014, no. 1, pp. 1-13, 2014.

[12] G. Xie, J. Ren, S. Marshall, H. Zhao, and H. J. I. A. Li, "A new cost function for spatial image steganography based on 2D-SSA and WMF," vol. 9, pp. 30604-30614, 2021.

[13] W. Su, J. Ni, X. Hu, and J. J. S. P. Huang, "New design paradigm of distortion cost function for efficient JPEG steganography," vol. 190, p. 108319, 2022.

[14] G. Xu, H.-Z. Wu, and Y. Q. Shi, "Ensemble of CNNs for steganalysis: An empirical study," in *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, 2016, pp. 103-107.

[15] G. Xu, H.-Z. Wu, and Y.-Q. J. I. S. P. L. Shi, "Structural design of convolutional neural networks for steganalysis," vol. 23, no. 5, pp. 708-712, 2016.

[16] B. Karlik, A. V. J. I. J. o. A. I. Olgac, and E. Systems, "Performance analysis of various activation functions in generalized MLP architectures of neural networks," vol. 1, no. 4, pp. 111-122, 2011.

[17] B. Li, Z. Li, S. Zhou, S. Tan, X. J. I. T. o. I. F. Zhang, and Security, "New steganalytic features for spatial image steganography based on derivative filters and threshold LBP operator," vol. 13, no. 5, pp. 1242-1257, 2017.

[18] J. Fridrich, J. J. I. T. o. I. F. Kodovsky, and Security, "Rich models for steganalysis of digital images," vol. 7, no. 3, pp. 868-882, 2012.

[19] B. Li, W. Wei, A. Ferreira, and S. J. I. S. P. L. Tan, "ReST-Net: Diverse activation modules and parallel subnets-based CNN for spatial image steganalysis," vol. 25, no. 5, pp. 650-654, 2018.

[20] S. Zhou, W. Tang, S. Tan, and B. Li, "Content-adaptive steganalysis via augmented utilization of selection-channel information," in *International Workshop on Digital Watermarking*, 2018, pp. 261-274: Springer.

[21] J. Lu, G. Zhou, C. Yang, Z. Li, and M. J. I. A. Lan, "Steganalysis of content-adaptive steganography based on massive datasets pre-classification and feature selection," vol. 7, pp. 21702-21711, 2019.

[22] F. Liu, X. Yan, and Y. J. I. A. Lu, "Feature selection for image steganalysis using binary bat Algorithm," vol. 8, pp. 4244-4249, 2019.

[23] T. Pevny, P. Bas, J. J. I. T. o. i. F. Fridrich, and Security, "Steganalysis by subtractive pixel adjacency matrix," vol. 5, no. 2, pp. 215-224, 2010.

[24] M. Huang, "Statistical Steganalysis of Images," Doctoral dissertation, Purdue University Graduate School, 2019.

[25] Z. Jin, Y. Yang, Y. Chen, and Y. J. I. J. o. D. S. N. Chen, "IAS-CNN: Image adaptive steganalysis via convolutional neural network combined with selection channel," vol. 16, no. 3, p. 1550147720911002, 2020.

[26] Z. Xiang, J. Sang, Q. Zhang, B. Cai, X. Xia, and W. J. I. A. Wu, "A new convolutional neural network-based steganalysis method for content-adaptive image steganography in the spatial domain," vol. 8, pp. 47013-47020, 2020.

[27] F. Liu, X. Zhou, X. Yan, Y. Lu, and S. J. M. Wang, "Image Steganalysis via Diverse Filters and Squeeze-and-Excitation Convolutional Neural Network," vol. 9, no. 2, p. 189, 2021.

[28] P. Bas, T. Filler, and T. Pevný, "" Break our steganographic system": the ins and outs of organizing BOSS," in *International workshop on information hiding*, 2011, pp. 59-70: Springer.

[29] B. Pfitzmann, "Information hiding terminology," in *Information Hiding: First International Workshop Proceedings, 1996*, 1996: Springer.

[30] T. Morkel, J. H. Eloff, and M. S. Olivier, "An Overview of Image Steganography" proceedings of the fifth annual Information Security South Africa Conference ,2005, vol. 1, no. 2, pp. 1-11.

[31] R. K. Sheth, R. M. J. I. J. O. C. E. Tank, and Sciences, "Image steganography techniques," vol. 1, no. 2, pp. 10-15, 2015.

[32] M. Chaumont, "Deep learning in steganography and steganalysis," in *Digital Media Steganography*: Elsevier, 2020, pp. 321-349.

[33] V. Hajduk and D. Levický, "Accelerated cover selection steganography," in *2017 27th International Conference Radioelektronika (RADIOELEKTRONIKA)*, 2017, pp. 1-4: IEEE.

[34] A. H. Ouda and M. R. El-Sakka, "A step towards practical steganography systems," in *International Conference Image Analysis and Recognition*, 2005, pp. 1158-1166: Springer.

[35] T. Filler, J. Judas, and J. Fridrich, "Minimizing embedding impact in steganography using trellis-coded quantization." in *Media forensics and security II*, vol. 7541, p. 754105, 2010.

[36] T. Pevný, T. Filler, and P. Bas, "Using high-dimensional image models to perform highly undetectable steganography," in *International Workshop on Information Hiding*, 2010, pp. 161-177: Springer.

[37] J. Kodovský and J. Fridrich, "On completeness of feature spaces in blind steganalysis," in *Proceedings of the 10th ACM Workshop on Multimedia and Security*, 2008, pp. 123-132.

[38] J. Fridrich and T. Filler, "Practical methods for minimizing embedding impact in steganography," in *Security, Steganography, and Watermarking of Multimedia Contents IX*, 2007, vol. 6505, p. 650502: International Society for Optics and Photonics.

[39] M. Yedroudj, "Steganalysis and steganography by deep learning," Université Montpellier, 2019.

[40] Ker, A.D., Bas, P., Böhme, R., Cogranne, R., Craver, S., Filler, T., Fridrich, J., and Pevný, T., "Moving steganography and steganalysis

from the laboratory into the real world," in *Proceedings of the first ACM workshop on Information hiding and multimedia security*, 2013, pp. 45-58.

[41] T. Filler, J. Judas, J. J. I. T. o. I. F. Fridrich, and Security, "Minimizing additive distortion in steganography using syndrome-trellis codes," vol. 6, no. 3, pp. 920-935, 2011.

[42] A. Winkler, "Advances in Syndrome Coding based on Stochastic and Deterministic Matrices for Steganography," PhD, Technischen Universität Dresden, 2011.

[43] J. Li, X. Luo, Y. Zhang, P. Zhang, C. Yang, F. J. D. C. Liu, "Extracting embedded messages using adaptive steganography based on optimal syndrome-trellis decoding paths," Digital Communications and Networks, 2021 Sep 22.

[44] M. Boroumand, M. Chen, J. J. I. T. o. I. F. Fridrich, and Security, "Deep residual network for steganalysis of digital images," vol. 14, no. 5, pp. 1181-1193, 2018.

[45] J. Fridrich and J. Kodovský, "Steganalysis of LSB replacement using parity-aware features," in *International Workshop on Information Hiding*, 2012, pp. 31-45: Springer.

[46] J. Kodovsky, *Steganalysis of digital images using rich image representations and ensemble classifiers*. State University of New York at Binghamton, 2012.

[47] T. Sharp, "An implementation of key-based digital signal steganography," in *International Workshop on Information Hiding*, 2001, pp. 13-26: Springer.

[48] V. Holub and J. Fridrich, "Designing steganographic distortion using directional filters," in *2012 IEEE International workshop on information forensics and security (WIFS)*, 2012, pp. 234-239: IEEE.

[49] W. Tang, H. Li, W. Luo, and J. Huang, "Adaptive steganalysis against WOW embedding algorithm," in *Proceedings of the 2nd ACM workshop on Information hiding and multimedia security*, 2014, pp. 91-96.

[50] V. Sedighi, R. Cogranne, J. J. I. T. o. I. F. Fridrich, and Security, "Content-adaptive steganography by minimizing statistical detectability," vol. 11, no. 2, pp. 221-234, 2015.

[51] S. J. J. o. g. r. i. c. s. Bhattacharyya, "A survey of steganography and steganalysis technique in image, text, audio and video as cover carrier," vol. 2, no. 4, 2011.

[52] J. Kodovsky, J. Fridrich, and V. Holub, "On dangers of overtraining steganography to incomplete cover model," in *Proceedings of the*

thirteenth ACM multimedia workshop on Multimedia and security, 2011, pp. 69-76.

[53] P. Wang, Z. Wei, and L. J. S. C. Xiao, "Spatial rich model steganalysis feature normalization on random feature-subsets," vol. 22, no. 6, pp. 1981-1992, 2018.

[54] J. Ye, J. Ni, Y. J. I. T. o. I. F. Yi, and Security, "Deep learning hierarchical representations for image steganalysis," vol. 12, no. 11, pp. 2545-2557, 2017.

[55] I. Lubenko, "Towards robust steganalysis: binary classifiers and large, heterogeneous data," 2013.

[56] V. Holub, J. Fridrich, and T. Denemark, "Random projections of residuals as an alternative to co-occurrences in steganalysis," in *Media Watermarking, Security, and Forensics 2013*, 2013, vol. 8665, p. 86650L: International Society for Optics and Photonics.

[57] J. Kodovský and J. Fridrich, "Steganalysis of JPEG images using rich models," in *Media Watermarking, Security, and Forensics 2012*, 2012, vol. 8303, p. 83030A: International Society for Optics and Photonics.

[58] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, B. J. I. I. S. Scholkopf, and t. applications, "Support vector machines," vol. 13, no. 4, pp. 18-28, 1998.

[59] A. Krizhevsky, I. Sutskever, and G. E. J. A. i. n. i. p. s. Hinton, "Imagenet classification with deep convolutional neural networks," vol. 25, pp. 1097-1105, 2012.

[60] I. Tabian, H. Fu, and Z. J. S. Sharif Khodaei, "A convolutional neural network for impact detection and characterization of complex composite structures," vol. 19, no. 22, p. 4933, 2019.

[61] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[62] L. Lu, Y. Zheng, G. Carneiro, L. J. A. i. c. v. Yang, and p. recognition, "Deep learning and convolutional neural networks for medical image computing," vol. 10, pp. 978-3, 2017.

[63] K. Jarrett, K. Kavukcuoglu, M. A. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?," in *2009 IEEE 12th international conference on computer vision*, 2009, pp. 2146-2153: IEEE.

[64] J. Feng, X. He, Q. Teng, C. Ren, H. Chen, and Y. J. P. R. E. Li, "Reconstruction of porous media from extremely limited information using conditional generative adversarial networks," vol. 100, no. 3, p. 033308, 2019.

[65] C. Nwankpa, W. Ijomah, A. Gachagan, and S. J. a. p. a. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," 2018.

[66] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, 2013, vol. 30, no. 1, p. 3: Citeseer.

[67] N. Aljaafari, "Ichthyoplankton classification tool using Generative Adversarial Networks and transfer learning," 2018.

[68] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International conference on artificial neural networks*, 2010, pp. 92-101: Springer.

[69] M. Lin, Q. Chen, and S. J. a. p. a. Yan, "Network in network," 2013.

[70] J. Bjorck, C. Gomes, B. Selman, and K. Q. J. a. p. a. Weinberger, "Understanding batch normalization," 2018.

[71] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448-456: PMLR.

[72] N. Buduma and N. Locascio, *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms*. " O'Reilly Media, Inc.", 2017.

[73] C. Heipke and F. J. G.-s. I. S. Rottensteiner, "Deep learning for geometric and semantic tasks in photogrammetry and remote sensing," vol. 23, no. 1, pp. 10-19, 2020.

[74] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. J. T. j. o. m. l. r. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," vol. 15, no. 1, pp. 1929-1958, 2014.

[75] W. J. J. o. F. R. M. Zhang, "Machine learning approaches to predicting company bankruptcy," vol. 6, no. 04, p. 364, 2017.

[76] D. E. Rumelhart, G. E. Hinton, and R. J. J. n. Williams, "Learning representations by back-propagating errors," vol. 323, no. 6088, pp. 533-536, 1986.

[77] Y. Chen, S. Biookaghazadeh, and M. Zhao, "Exploring the capabilities of mobile devices in supporting deep learning," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 2019, pp. 127-138.

[78] D. P. Kingma and J. J. a. p. a. Ba, "Adam: A method for stochastic optimization," 2014.

[79] J. Duchi, E. Hazan, and Y. J. J. o. m. l. r. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," vol. 12, no. 7, 2011.

[80] T. Tieleman and G. J. C. N. N. M. L. Hinton, "Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning," 2012.

[81] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., and Cai, J.J.P.R.: 'Recent advances in convolutional neural networks', 2018, 77, pp. 354-377

[82] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249-256: JMLR Workshop and Conference Proceedings.

[83] R. C. Eberhart, Y. Shi, and J. Kennedy, *Swarm intelligence*. Elsevier, 2001.

[84] P. Moradi and M. J. E. A. o. A. I. Rostami, "A graph theoretic approach for unsupervised feature selection," vol. 44, pp. 33-45, 2015.

[85] T. N. Lal, O. Chapelle, J. Weston, and A. Elisseeff, "Embedded methods," in *Feature extraction*: Springer, 2006, pp. 137-165.

[86] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, 1995, vol. 4, pp. 1942-1948: IEEE.

[87] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*, 1998, pp. 69-73: IEEE.

[88] S. Wang, F. Zhou, and F. J. I. J. o. O. E. Wang, "Effect of Inertia Weight ω on PSO-SA Algorithm," vol. 9, 2013.

[89] A. P. Engelbrecht, *Computational intelligence: an introduction*. John Wiley & Sons, 2007.

[90] Y. Xiao, Y. Wang, and Y. J. E. Sun, "Reactive power optimal control of a wind farm for minimizing collector system losses," vol. 11, no. 11, p. 3177, 2018.

[91] El-Shorbagy, Mohammed A., and Aboul Ella Hassanien. "Particle swarm optimization from theory to applications." *International Journal of Rough Sets and Data Analysis (IJRSDA)* 5, no. 2 (2018): 1-24.

[92] Abd Allah, A. Mousa, and Mohamed A. El-Shorbagy. "Enhanced particle swarm optimization based local search for reactive power compensation problem." (2012).

[93] El-Shorbagy, M.A., 2015. Weighted Method Based Trust Region-Particle Swarm Optimization for Multi-Objective Optimization. *American Journal of Applied Mathematics*, *3*(3), pp.81-89.

[94] Meza, J., Espitia, H., Montenegro, C., Giménez, E. and González-Crespo, R., 2017. MOVPSO: vortex multi-objective particle swarm optimization. *Applied Soft Computing*, *52*, pp.1042-1057.

[95] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm Algorithm," in *1997 IEEE International conference on systems, man, and cybernetics. Computational cybernetics and simulation*, 1997, vol. 5, pp. 4104-4108: IEEE.

[96] M. S. Mohamad, S. Omatu, S. Deris, and M. J. I. T. o. i. T. i. B. Yoshioka, "A modified binary particle swarm optimization for selecting the small subset of informative genes from gene expression data," vol. 15, no. 6, pp. 813-822, 2011.

[97] X. Zhu and H. Wang, "A new inertia weight control strategy for particle swarm optimization," in *AIP Conference Proceedings*, 2018, vol. 1955, no. 1, p. 040095: AIP Publishing LLC.

[98] C. Yang, W. Gao, N. Liu, and C. J. A. S. C. Song, "Low-discrepancy sequence initialized particle swarm optimization Algorithm with high-order nonlinear time-varying inertia weight," vol. 29, pp. 386-394, 2015.

[99] V. V. Naumovich and V. Vlamimir, "Statistical learning theory," ed: Wiley New York, 1998.

[100] D. D. Shankar and P. K. Upadhyay, "Steganalysis of Very Low Embedded JPEG Image in Spatial and Transform Domain Steganographic Scheme Using SVM," in *Innovations in Computer Science and Engineering*: Springer, Singapore, 2020, pp. 405-412.

[101] M. Hajmeer and I. J. F. M. Basheer, "Comparison of logistic regression and neural network-based classifiers for bacterial growth," vol. 20, no. 1, pp. 43-55, 2003.

[102] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013.

[103] W. Vach, R. Roßner, and M. Schumacher, "Neural networks and logistic regression: Part II," vol. 21, no. 6 %J Comput. Stat. Data Anal., pp. 683–701, 1996.

[104] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[105] S.-C. Hsu, I.-C. Chen, C.-L. J. J. o. I. S. Huang, and Engineering, "Image Classification Using Naive Bayes Classifier With Pairwise Local Observations," vol. 33, no. 5, 2017.

[106] L. J. M. l. Breiman, "Random forests," vol. 45, no. 1, pp. 5-32, 2001.

[107] S. J. d. c. Awasthi, "Random Forests in Machine Learning: A Detailed Explanation," 2020.

[108] S. Duque and M. N. J. P. C. S. bin Omar, "Using data mining Algorithms for developing a model for intrusion detection system (IDS)," vol. 61, pp. 46-51, 2015.

[109] Powers, David MW. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation." *arXiv preprint arXiv:2010.16061* (2020).

[110] D. M. J. a. p. a. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," 2020.

[111] Hand, David J., Peter Christen, and Nishadi Kirielle. "F*: an interpretable transformation of the F-measure." *Machine Learning* 110, no. 3 (2021): 451-456.

المستخلص

إخفاء البيانات وتحليل الاخفاء هما تقنيتان متناقضتان. إخفاء البيانات هي تقنية لتضمين المعلومات السرية في محتوى الوسائط المتعددة لجعلها غير قابلة للكشف، بينما تقنية معرفة ما إذا كان هناك معلومات سرية مخبأة في الصور تعرف باسم تحليل الاخفاء. بما أن طرق الاخفاء في الصور المتكيفة مع المحتوى تدمج المعرفة على نحو تكيفي في المناطق الغنية بالنسيج والقائمة على التشويه، فإن التطور الحاصل في هذه الطرق جعل أحدث نُهج تحليل الاخفاء غير قادرة على التمييز بين الصور التي تحتوي على المعلومات المخفية من تلك الخالية منها, أو يتم اكتشافها بدقة اقل، مما يجعل ذلك تحدياً مستمراً.

نظراً للأداء المميز الذي حققته الشبكات العصبية التلافيفية في مجال معالجة الصور، فقد ركز عدد متزايد من الباحثين على تطوير أساليب لتحليل الاخفاء بالاعتماد على هذه الشبكات. على أي حال، تهدف هذه الأطروحة إلى تطوير نظام يعتمد في استخلاص الميزات على الشبكة العصبية التلافيفية واختيار مجموعة الميزات الأفضل من بينها لغرض تحليل الاخفاء في الصور لطرق الاخفاء تكيفية المحتوى.

بعد تحضير مجموعات بيانات طرق الاخفاء، تستند الطريقة المقترحة على تطبيق مرشح مخلفات الضوضاء على مجموعات البيانات لكبت محتويات الصورة، ثم استخلاص ميزات الصور باستخدام الطريقة المقترحة والقائمة على استخدام الشبكة العصبية التلافيفية. بعد ذلك، يتم استخدام خوارزمية اسراب الطيور لاختيار أكثر الميزات فائدة والغنية بالمعلومات، وأخيراً، تدريب مصنف آلة المتجه الداعم باستخدام ميزات التدريب المختارة واستخدام نفس المصنف لمعرفة ما إذا كان هناك معلومات مخبأة في الصور باستخدام مجموعات الاختبار باعتبارها صوراً جديدة للتنبؤ بها.

لتقييم أداء الطريقة المقترحة، تم استخدام خمساً من طرق الإخفاء وكانت نتائج الدقة تتراوح من ٦٩,٨٠% الى ٨٥,٩٦% , من ٧١,٠٠% الى ٨٧,٥٣%, من ٧١,٢٢% الى ٨٥,١٢%, من ٦٠,٧٧% الى ٨٨,١٦%, وأخيرا من ٦٥,٣٩% الى ٨٧,٠٤% لطرق الاخفاء HILL , HUGO , MiPOD , و S-UNIWARD, على التوالي.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل- كلية تكنولوجيا المعلومات
قسم البرمجيات

تحليل الاخفاء في الصور لطرق الاخفاء تكيفية المحتوى بالاستناد الى الشبكة العصبية التلافيفية وآلة المتجه الداعم

أطروحة مقدمة

الى مجلس كلية تكنولوجيا المعلومات في جامعة بابل وهي جزء من متطلبات نيل درجة الدكتوراه فلسفة في تكنولوجيا المعلومات / برمجيات

من قبل

سعيد محمد هاشم ياسين

إشراف

أ. د. ضياء عبد الحسين جمعة

٢٠٢٢ م ١٤٤٣ هـ