Malicious Image Detection Using Convolutional Neural Network

Thesis Submitted By:
Ahsan Iqbal & 01-243172-038

Supervised By:
Dr. Samabia Tehseen

A dissertation submitted to the Department of Computer Science, Bahria University, Islamabad as a partial fulfillment of the requirements for the award of the degree of Masters in Computer Science

Session (2017-2019)

Bahria University
Discovering Knowledge

MS-13

Thesis Completion Certificate

Scholar's Name: AHSAN IQBAL Registration No. 31865

Programme of Study: MSCS

Thesis Title: Malicious Images Detection using Convolutional Neural Network

It is to certify that the above student's thesis has been completed to my satisfaction and, to my belief, its standard is appropriate for submission for Evaluation. I have also conducted plagiarism test of this thesis using HEC prescribed software and found similarity index at 8 % that is within the permissible limit set by the HEC for the MS/MPhil degree thesis.
I have also found the thesis in a format recognized by the BU for the MS/MPhil thesis.

Principal Supervisor's Signature: _____

Date: 26th June, 2019 **Name:** Dr Samabia Tehsin

Bahria University
Discovering Knowledge

MS-14A

Author's Declaration

I, AHSAN IQBAL hereby state that my MS thesis titled "Malicious Images Detection using Convolutional Neural Network."

is my own work and has not been submitted previously by me for taking any degree from this university

Bahria University, Islamabad (Name of University)

or anywhere else in the country/world.

At any time if my statement is found to be incorrect even after my Graduate the university has the right to withdraw/cancel my PhD degree.

Name of scholar: AHSAN IQBAL
Date: 26th June, 2019

Bahria University
Discovering Knowledge

MS-14B

Plagiarism Undertaking

I, solemnly declare that research work presented in the thesis titled "Malicious Images Detection using Convolutional Neural Network" is solely my research work with no significant contribution from any other person. Small contribution / help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and Bahria University towards plagiarism. Therefore I as an Author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred / cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of PhD degree, the university reserves the right to withdraw / revoke my PhD degree and that HEC and the University has the right to publish my name on the HEC / University website on which names of students are placed who submitted plagiarized thesis.

Student / Author's Sign:

Name of the Student:
AHSAN IQBAL

Abstract

In previous era, malware attacks have achieved serious heights. As information technology field strengthens, the activities of cyber-criminals are also updated. Cyber-criminals always look for those methods to attack which are not much suspicious. Attackers started to use approaches like steganography to conceal the scripts. With the wide use of images on social media and other platforms like World Wide Web (WWW), attackers started to embed the malwares in images. With the growth of malware attacks through images, it is high time to introduce a technique which would detect the malicious images. Proposed study aims the detection of images which are concealed with different scripts. We used a dataset of JPEGs, containing 1100 malicious and 1100 benign images to employ the detection method. Our method of malicious image detection would help everyone to prevent the malware attacks which are carried through images.

Acknowledgments

My heart is replete with thankfulness to Almighty Allah, who granted me with strength and courage to complete my thesis. The coaching and guidance given by my supervisor Dr. Samabia Tehseen, throughout my thesis was sumptuous. I want to command Dr. Rizwan Abu Ahmed for sharing valuable suggestions and for providing guidance to complete my thesis. I thank Dr. Sumaira Kausar and Dr. Faisal Bashir for providing the valuable guidelines during meetings. A special thanks to CRC lab for providing the dataset. I would like to thank my friends Wajaht Abbasi, Zeeshan Liaqat and my cousin Danial Arshad for their valuable support toward my thesis.

May Allah reward my family, my professors and my friends, and have mercy on them who provided me with the support which I required during my thesis.

AHSAN IQBAL
Bahria University Islamabad, Pakistan

June, 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today there are many ways that attackers are using to harm networks, systems and databases etc. While evolving the old age techniques into advance forms, the malware activities are also get updated. Attackers achieve different goals by attacking systems and networks; those goals may include to steal important information from networks or databases and sometimes to get remote access of the systems. There are some benefits for which those attacking activities are led by the cyber-criminals. Advancement in technology and different means of attacks are updated in parallel. With new discoveries and innovations in technology, attackers always modernize their methods to attack.

Attackers have found a new way to attack which involves images, as images are known to be harmless. It is learnt that malicious codes are concealed into different formats of images by using different techniques. Online Social Networks (OSN) [21] is a new platform targeted by cyber-criminals for malware attacks by using malicious images. A lot of content is shared on social media across the world based on images which makes attacker’s job easy. Payloads are easily transferred into systems since social media users are oblivious about malware attacks based on malicious images.

Payloads can be greatly damaging for systems and databases because attackers can easily get remote access of systems by execution of those payloads. Now it is need of the hour to prevent remote access of systems and loss of information by applying some new detection methods. Limited work has been done for detection of malicious images. It is required to introduce a prolific method by using deep learning for detection of malicious images.

Different techniques are used to conceal malicious codes in images. Now Steganography is also used to conceal payloads in images. It is a technique to hide a message or a file in another file. This technique can be applied on many file formats, best known for images and audios. For images, you can hide an image in another image without anyone knowing. Only one image will be visible while other will be hidden in it. This process also losses little information of those images. Effect of those is barely discernible for human eye. Now it has got attention of the attackers to hide malicious codes in images.

It is learnt that the most effective cyber-criminal activities [4, 2] use JPEG image formats to carry the script. Two reasons to that:

1. JPEGs are mostly used format across Word Wide Web (WWW).

2. JPEGs are easy to embed with malicious content without anyone’ s suspicion.

JPEGs header can contain different scripts e.g. JavaScript and PHP commands etc. Different steganography related methods (not publically available) are used to embed scripts in JPEGs. JPEGs comprise of different segments in the header. Segments are represented by the markers which contain Hexadecimal values, section 3.1 describes about structure of JPEGs. The script is inserted in those hexadecimal values.

For the detection purposes of malicious JPEGs, the hex values of dataset are read in Byte form. Some of the Number Systems are discussed in section 3.2 along Base64 Encryptions technique because different browsers e.g. Google Chrome and other targets read images in Base64 form and decode the image content to display it, figure 1.1 shows. Hence Base64 is also used to encrypt the scripts to make it compatible to carry malware scripts in images.

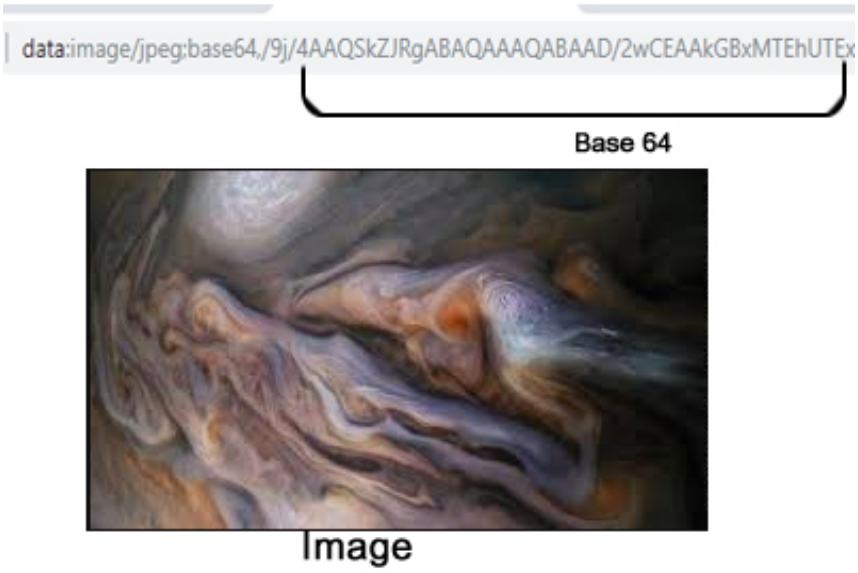

Figure 1.1: Google Chrome Reading JPEGs in Base64

That byte form of Hex is converted to 8-bit grayscale image. A deep learning based technique named as Convolutional Neural Network (CNN) is employed on grayscale images for detection of malicious image. Our problem is a two class problem comprising of Benign class and Malicious class. Benign class, as name suggests is class to represent benign images and Malicious class is for malicious images which carry the scripts.

1.1 Motivation and Problem Description

There are numerous new techniques used to embed malicious script in different file types. Images are most preferred file type for attacks because they seem to be harmless and non-risky. Malicious codes are embedded in images by using different techniques [4, 2] e.g. Steganography to attack social media networks and other systems. Vulnerability of systems and networks can be easily exploited by using malicious images which could result in important information loss or system damage. It is needed to study and develop a tool for malicious image detection.

1.2 Research Contribution

1. Designed CNN based classifier for malicious image detection.

2. Documented the image format and structure of the image format which is used for malicious attacks.

3. Explored different techniques for incorporating malicious codes in images e.g. Stegosploit.

4. Cross validated and well tested the results of research and compiled and analyzed in the final documentation.

1.3 Thesis Organization

Thesis consists of 6 chapters. This chapter is followed by chapter 2 which includes detailed literature of background knowledge and script embedding techniques. In chapter 3, the structure of image format and number systems with encryption technique are included. Chapter 3 is followed by chapter 4, where methodology of proposed study is discussed with the detailed discussion for data preparation and description about data is added. Chapter 5 includes results and discussion of experiments performed. At last chapter 6 has conclusion of our research work.

Chapter 2

Literature Review

2.1 Malware classification in other file types

There is a lot of research work on malware classification in different file types. Unknown malwares types aren't classified previously but Lui et al. [25] proposed a method for classification of malware among different malware families and clustering of new detected malware types. Proposed method is composed of three steps; first one is feature extraction, second is selection of decision making and third is new malware detection. For feature extraction 3 methods were used, e.g, gray-scale images, import function and Opcode n-gram. For grayscale image, malware files were converted to binary files by using interactive disassembler (IDA). Content of binary file were divided into 8 bit units. Then that file was represented as grayscale image. For Opcode n-gram, IDA pro was used, which exploits the coding function flow of malware files. They produced control graph, which helps for texture feature extraction. They combined 3-gram and CFG to extract feature for malware. They also used counts of dll files which are being called in import function of Windows and they used those counts as feature for malware. Information gain was used to check the effectiveness of features and to reduce the number of features used. Shared nearest neighbor (SNN) was used because of its performance with high dimensional features. Dataset was collected from VX Heaven, ESET and NOD32. 21,740 instances of dataset were used which were divided into 19,740 training and 2,000 test samples. 7 different classifiers were used for accuracy performance based on gray scale image, Opcode n-gram and combined features. SNN method was used to cluster

those malware. Proposed system achieved 98.9% accuracy to classify and for new malware detection proposed system was limited to 86.7% accuracy.

Previous techniques were used to detect malware by using code analysis but with obfuscation in codes it was difficult to detect malwares. Ajit el al. [20] proposed a system which classify android apks as malware or benign by visualizing. They performed training on random forest (RF), decision tree (DT), K nearest neighbor. Performance was measured based on precision, recall and accuracy. Data set had 246 samples among which they had 108 benign apks and remaining 138 were malware apks. DT had worst results as it had 76% average accuracy. However, RF had better performance, producing 86% accuracy for all formats' feature set and it had 91% for features based on gray-images.

2.2 Malicious code embedding techniques

2.2.1 Stegosploit

Malicious code embedding techniques in images are mentioned by cyber security researcher in different articles and conferences' presentations. A few are discussed below. Images are known for being non-risky and non-harmful. It is learnt that now images are also targeted by the attackers for malicious code execution. An algorithm can leak information on system or it can damage a system. Sumail Shah [4] explained how a JavaScript code can be embedded in image and that can be executed through a browser. Shah named the technique "Stegosploit" referring from word Steganography. He described how a malware attacker can attack and send or receive information. In this technique malicious code is concealed in pixel data. A HTML5's <canvas> tag is used for this purpose which reads image data as JavaScript code. Browser reads a jpg image and executes pixels data where malicious code is decoded and then that code is executed. He referred JavaScript code within image as "IMAJS".

There are different image formats which support Stegosploit. For different image formats, there are different ways for concealing malicious codes. APP0 segment of JPEG supports malware code insertion. For PNG it is stored in tEXt chunk. Script is inserted at the end of image in BMP and GIF formats, which is then refered by HTML <img> tag and script runs when malicious image is called in <script> tag.

2.2.2 Comment segment injection

Murray [2] explained a technique where he embedded asp.net code to a JPEG file. He basically performed it in different way than Shah did. He injected another comment

field in JPEG metadata. Comment field contained all malicious code from the attacker. In this demo for execution of code, he changed JPEG extension with .aspx extension which is asp.net’ s extension.

2.2.3 ZeusVM

Jerome Segura [1] wrote about the case where he found malware in JPG file. It is said that ZeusVM was used with steganography to implement that malware in JPG. Image caring malware was a common image he got its sample images from Google Search. He analyzed original and malicious image by converting JPG to bitmap.

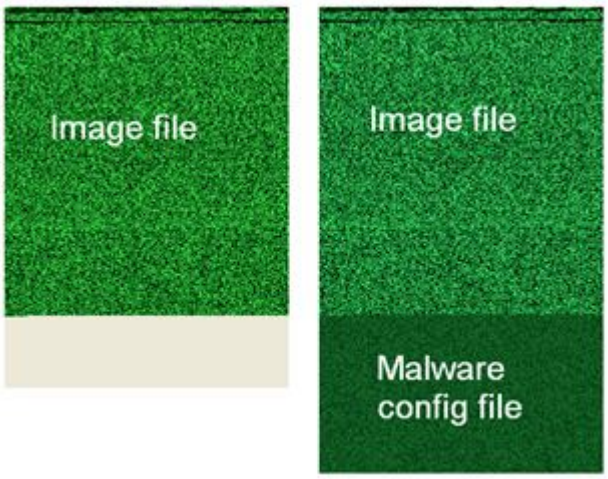

Figure 2.1: Image in bitmap form

He analyzed the malicious image in hexadecimal viewer. The segmented code was visible at the end of image pixel code.

Over social media it is now easy to attack through images and get remote access of the system. In a study, Rakesh et al. [21] suggested framework architecture to detect malicious code presence in images. It is studied that Online Social Networks (OSNs) are now attack oriented places where payload can be easily transmitted and executed by using malicious image. In this research some Steganography tools were mentioned along their respective detecting Steganalysis tools. Proposed architecture is based on 3 phases. First phase is Steganalysis of captured images; image is analyzed with techniques like Histogram analysis attack and Chi-square attack. Then it has second phase, if first phase indicate presence of malicious code, metadata is extracted to get to know about stego-image and it is analyzed whether concealed file is executable or not by considering .exe, .dll and other extensions. In third phase file is unpacked so that code could be analyzed and on those bases real-time antiviruses would be updated.

Malicious attacks through android applications increased with increase in the usage of android operating system. Shikha et al. [12] proposed a method where JPEG/PNG image formats were used to carry malicious codes in form of android application. They used different combination of concatenation (for hiding code by using UNIX 'cat' command), obfuscation (for code transformation by using ProGuard), cryptography (for code transformation by using XOR encryption) and Steganography (for hiding code by using Outguess 0.1 Algorithm) to embed malicious codes in image resource of android apks. They concatenated malicious code at the end of pixel data. In another method they used Steganography technique e.g. Least significant Bit (LSB) to embed malicious codes in images. Resources were split and decrypted and then extracted malware app was invoked by using DexClassLoader and Reflection. Detection of malware was validated across 10 malware detection Software. Only one instance of concatenation implied resource was detected by only one android based anti-virus.

2.3 Steganalysis

Steganalysis is used to detect Steganography in images. Dong-Hyun et al. [18] proposed a method based on Deep Learning for Steganalysis of Least Significant Bit (LSB). They used Convolutional Neural Network (CNN) to detect Stego-images based on BOSS and SIPI databases. Highly pass filter (HPF) used for noise extraction. Model had 2 convolutional layers and 2 fully connected layers. They collected 10,000 cover (original) images and 20,000 Stego-images. 80% of images were used for the training and 20% for testing. Method produced the results with 90% accuracy for LSB Stego-images with different keys and 98% accuracy for LSB Stego-images with same key.

Jian et al. [30] proposed a method for Steganalysis which is based on CNN. Truncated linear unit (TLU) activation function was used. For evaluation 3 Steganography algorithms were used, which are known as S-UNIWARD, HILL, and WOW. With activation functions ReLU and TLU, first convolutional layers' different initialization strategies were analyzed to evaluate their performance. Datasets used for training were BOWS2, AUG (augmented data by using BOWS2 and BOSS datasets) and BOSS and for testing BOSS-test was used. TLU's another version Selection-Channel-Aware SCA-TLU was also used. Proposed model was compared with hand crafted features set SRM and Selection-Channel-Aware maxSRM2d. It was also learnt that more the payload ratio, better the detection performance of suggested model gets. The table 2.1 gives short overview of literature.

Table 2.1: Related research work done

Author	Methodology	Dataset	Limitations/ Remarks
Jian et al. 2017 [30]	Steganalysis by using CNN on Stego-Images Based On S-UNIWARD, HILL, and WOW Steganography technique	BOWS2 and BOSS	Doesn't detect malicious images by Steganalysis.
Dong-Hyun et al. 2017 [18]	Steganalysis by using CNN on Stego-Images Based On LSB	BOSS and SIPI	Doesn't detect malicious images by Steganalysis.
Shikha et al. 2018 [12]	Research shows some ways to conceal codes in image resource of android apks by using concatenation and Steganography with combination of obfuscation and cryptography. Then validated detection performance of different antiviruses	Dataset was augmented by researchers, which isn't publically available	Doesn't detect malicious images by Steganalysis.
Lui et al. 2017 [25]	Proposed system classifies malware among known classes and detects new malware families and cluster unknown malware by using SNN.	Dataset was collected from VX Heaven, ESET and NOD32.	Doesn't detect malicious images by Steganalysis.
Ajit el al. 2016 [20]	Proposed method classifies android apks to malicious or benign, based on visual features	They collected dataset from different sources. Dataset is not publically available	Doesn't detect malicious images by Steganalysis.

2.4 Watermarking

The protection of data authorization is necessary due to increase of attackes which manipulates data. To prevent unauthorized use of images, the area of digital watermarking is recognized to protect the copyright information. B. Kaur et al. [17] presented a scheme of steganography for hiding image in discrete cousin transformation (DCT) domain to provide resistance to image processing attacks. To embed the watermarks, mid frequency band of DCT was used because watermark information is not scattered to most visual parts of the image. They used 512x512 grayscale 'Lena' with logo of 64x64 grayscale image of copyright for watermark. Various types of noises like Gaussian noise, salt and pepper noise and speckle noise were subjected on cover image and result for each type of noise for maximum extent that can be tolerated, results were presented. Proposed method explored the DCT domain for watermarking over the gray scale images.

Images are manipulated for unauthorized use which losses the authenticity of an image, Ching-Y. Lin et al. [24] proposed a system which is used to ensure authenticity

of image. It distinguishes malicious manipulation from lossy compression of JPEG. Invariance properties (which can be preserved during lossy compression of JPEG) are used. Image authenticator was tested on different manipulations of "Lenna" image.3 experiments performed with different compression ratios i.e 9:1, 6:1. Performance of the system was analyzed on the probability basis of miss and success. It is noted that as JPEG quality factor increases the median values of miss, decreases. And as manipulated values goes away from 0 in positive or negative direction the median values of probability success decreases. Proposed method distinguishes lossy compression of JPEG from other malicious manipulations.

Many tools and methods for malware detection and classification are proposed for other file types, where malware after detection is classified into existing malware families. In some studies, it is explained about methods to embed malicious codes in images by using different techniques. But solutions for malware detection in images have not been discussed.

Chapter 3

Related Concepts

It is important to have proper knowledge of JPEGs before going further into depth of JPEGs and detection process. In this chapter the structure of JPEGs and the Number System & Encryption are discussed.

3.1 JPEG Structure

JPEG is a digital image file type. It is used for lossy compression of digital images. To carry JPEG stream, multiple file formats are used which includes JPEG/SPIFF (Still Picture Interchange File Format), JPEG/CIFF, JPEG/Exif(Exchangeable Image File Format) and JPEG/JFIF (JPEG Interchangeable File Foramat) [3].

Most common file formats for JPEG are JPEG/Exif and JPEG/JFIF. These file formats are used for different purposes. Both file formats follow approximately similar structure. The major difference of these file formats have is in the Application segment. Exif and JFIF differences are following

3.1.1 Exif

- Exchangeable image file format.
- Used by digital cameras (including smartphones), scanners.
- APPn stores information about camera state (shuter, white balance).

3.1.2 JFIF

- JPEG File Interchange Format.
- Used for software saving and for word wide web.
- APPn stores information such as copyright and captions (IPTC text) and profile information for color management (icm data).

JPEG files has several segments, each of the segments contains different type of data. Segments are delimited by markers which are 2 byte codes. Markers are hexadecimal values which begin with 0xFF byte, followed by a byte which indicates the type of marker. Some markers have these two bytes and in some markers these two bytes are followed by high then low bytes, which indicate the length of marker-specific payload data. Entropy-coded data follows some of the markers, here padding are used, 0xFF byte indicate fill byte. This kind of padding is used for those markers which are following entropy-coded scan data. Entropy-coded scan data is compressed data of the JPEG [10].

A 0x00 byte is inserted by encoder after 0xFF byte if 0xFF occurs within entropy-coded data. This method is known as byte Stuffing. This prevents framing error. While decoding 0x00 are skipped by decoder within entropy-coded data. There could be independent chunks of entropy-coded data. To isolate these independent chunks, Reset markers are used to allow parallel decoding. These makers are indicated by 0xD0 to 0xD7 bytes. These markers belong to entropy-coded data.

Then there is SOI (start of image) marker which indicates the start of the image, this marker opens the file. 0xD8 is the byte used to indicate this marker. After this there is SOF0 and SOF2, start of frame markers. These markers are indicated by 0xC0 and 0xC2 byte respectively. In the jpeg another important markers are for the Huffman tables and Quantization tables. 0xC4 byte is used to indicate DHT (define Huffman table) and 0xDB byte is used for indicating DQT (define Quantization table) [10].

Intervals between RSTn markers are specified by DRI (define Restart interval) marker, which has fixed size of 4 bytes and indicated by 0xDD byte. RSTn markers aren’ t used if DRI marker is not present. For Top-to-Bottom scan of the image SOS (start of scan) marker is used. This marker identifies the slice of data it will have. It is followed by entropy-coded data. 0xDA byte is specified for this marker.

In JPEG most important marker type is APPn (Application). This segment indicates the file type of JPEG. For the JFIF, JPEG has APP0 marker and for Exif, JPEG has APP1, APP2 segments. For APPn segment byte value 0xEn is used. APP1 segment contains the metadata of JPEG image. APP0 segment stores picture dimensions and thumbnail information.

COM (Comment) segment is used to contain simple text comments. It is associated with 0xFE byte. EOI (End of Image) is the marker which indicates the end of the image. 0xD9 byte is used to indicate end of image [10]. Table 3.1 gives short overview of markers in JPEG structure.

Table 3.1: Markers in JPEG structure.

Short name	Bytes	Payload	Name
SOI	0xFF, 0xD8	none	Start Of Image
SOF0	0xFF, 0xC0	variable size	Start Of Frame (baseline DCT)
SOF2	0xFF, 0xC2	variable size	Start Of Frame (progressive DCT)
DHT	0xFF, 0xC4	variable size	Define Huffman Table(s)
DQT	0xFF, 0xDB	variable size	Define Quantization Table(s)
DRI	0xFF, 0xDD	4 bytes	Define Restart Interval
SOS	0xFF, 0xDA	variable size	Start Of Scan
RST*n*	0xFF, 0xD*n*(*n*=0..7)	none	Restart
APP*n*	0xFF, 0xE*n*	variable size	Application-specific
COM	0xFF, 0xFE	variable size	Comment
EOI	0xFF, 0xD9	none	End Of Image

3.2 Number System and Encryption Technique

It is important to understand various number systems, because in various methods the script is encoded or translated into another system which could help to reside the script inside the image. As the literature suggests [1, 2] that there are many methods where script is encoded and inserted into various parts of image, and to make our script supported by that image part, it is necessary to convert the script into a form which could be easily embedded into the image.

Some of the methods are explained below for number systems.

3.2.1 Binary

It is a number system where we have base (or radix) 2. 0 and 1 are the two symbols which are used to represent binary number system [22]. This number system is based on the positions. It uses power of 2 to determine the value. The bits which are left-most are known as MSB (Most Significant Bit), and the bits which are right-most are known as the LSB (Least Significant Bit). The arithmetic operations like addition, subtraction, division

and multiplication can be performed on this number system. Every symbol is denoted by a bit and any value can be represented by the sequence of bits [8].

3.2.2 Hexadecimal

This number system is of base 16. Sometimes this is denoted with hex word. There are 16 different symbols used for this number system [22]. It has first 10 values indexed like decimal numbers from 0 to 9, after decimal numbers it uses 6 alphabet values from A to F. Power of 16 is used to determine the values in this system. Hex is a positional number system which follows same most significant and least significant digits as binary system or any other system follows. This number system is used in transfer encoding Base16, where a byte of plain-text is broken into two 4-bit values and two digits of hex are used for that byte representation. 4 binary digits are used to represent one digit of hex and a byte can have values from 0000 0000 to 1111 1111 and its representation in hex is like 00 to FF [9]. In programming languages i.e. c language, the representation of hex involves prefix 0x or suffix. For example a value of hex is (2CA4)16 and in programming language it would denote the value as 0x2CA4, where 0x is prefix.

3.2.3 Octal

Base of octal number system is 8. In this system 0 to 7 symbols are used for representation of the system and the power of 8 determines the value [22]. Octal number system has standards of most significant and least significant digits same as the above two methods have. Modern computing platforms use 16, 32 or 64 bit words and on those platforms, 3 octal digits are represented by 1 byte and most significant digit of octal represents two binary digits [11]. Table 3.2 gives the representation of above explained number systems.

Table 3.2: Hex, Dec, Octal, Binary digits represntations.

Dec	Hex	Octal	Binary
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	8	10	1000
9	9	11	1001
10	A	12	1010
11	B	13	1011
12	C	14	1100
13	D	15	1101
14	E	16	1110
15	F	17	1111

3.2.4 Base64

Base64 is a system which uses 64 indexed values. Starting from 0 to 63, base64 uses capital alphabets from A to Z for first 26 representations of the indexed values, then it has small alphabet values from a to z for next 26 values. At the end it uses + and / symbol for 62 and 63 values' representation [26]. The table 3.3 shows base64 character along indexes.

Table 3.3: Base64 digits represntations.

Index	Char	Index	Char	Index	Char	Index	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

6 bits are used to represent each base64 digits. Four 6 bit base64 digits would be represented by 3 bytes or we can say 24 bits [7].

Encoding

Base64 is an algorithm which is widely used for cryptography purposes. Main purpose of cryptography is to convert the information into a variation which is not understandable and unreadable by unauthorized person. Base64 is basically used for binary to text encoding [26]. Base64 follows simple steps for encoding

- Convert text to ASCII code
- Convert ASCII value into binary values of 8 bit
- Combine the last 8 bits to 24 bits
- Produce 4 fractions of 24 bit into 6 bits
- Convert each 6 bit segment into a decimal value

- Convert decimal values into base64 digits w.r.t indexed values of base64

Important property of this binary to text transformation is that there is no any key or password involved [7].

ASCII

American Standard Code for Information Interchange is used for character encoding [5]. ASCII codes are used for text representation in computer systems. It is an English alphabet based encoding standard which uses seven bit integers to encode 128 characters. 95 characters of ASCII include uppercase alphabets A-Z, lowercase alphabets a-z, digits 0-9 and some printable punctuation symbols.

Table 3.4: ASCII values represntations.

Dec	Char	Dec	Char	Dec	Char	Dec	Char
0	NUL	32	Space	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	DEL

The extended ASCII codes were added which included index values from 128-255. The character encoding of extended ASCII is based on 8 bits. Extension further introduces 128 characters for European and Latin American languages [6].

3.3 ConvNet Architectures

CNN based models are composed of two segments, one is the convolutional base and the other is the classifier. We use the convolutional base of some model to learn the features

based on our dataset. And we feed those features to some classifier which is basically composed of fully connected layer. We can use pre-trained models for multiple purposes

- Training the entire model: we use the architecture of some pre-trained model and we train entire model on our dataset.
- Fine-tune a pre-trained model: we freeze some of the layers in convolutional base and fine-tune the high-level layers of network.
- Use CNN as feature extractor: we freeze convolutional base of a model and we pass the data through the network and feed the features to another classifier.

In our scenario dataset is small and it is different than the dataset of models which they have been trained on. We have used different ConvNet models and fine-tuned them and trained some of the layers in the architecture. For binary classification we fed the features to a sigmoid based classifier.

3.3.1 AlexNet

In ImageNet competition Alex Krizhevsky et al. [19] proposed a neural network architecture which is similar to LeNet-5 [23] architecture. Input to the proposed method has 227x227x3 dimensions which are then fed to convolutional layers where 96 filters of 11x11 are used with stride 1. Output of 55x55 is fed to pooling layer of 3x3 filters with stride 2, followed by couple Fully Connected layers with 4096 neurons for each layer. Then there is Fully Connected layer with 1000 neuron for 1000 classes with softmax function. Total parameters learnt are 60 Million. The activation function used in AlexNet is ReLU instead of tanh or sigmoid. Because ReLU is faster and produces the same accuracy with speeding up the process 5 times[23]. To reduce the problem of overfitting Droupout layer is used after each of the Fully Connected layer.

3.3.2 VGG

VGG architecture is introduced by Visual Geometry Group, which improves the AlexNet with help of using multiple small filters instead of large filter sizes [27]. Multiple 3x3 filters are used instead of 11 and 5 filter sizes in first two layers. In this architecture multiple small size filters increase the depth of the network with complex features but the cost of learning remains low. Only size of 3x3 filter is used with stride 1 in every Convolutional layer. And padding is 2x2 with stride equal to 1. Parameters learnt are 138 million.

3.3.3 Inception V1, V2, V3

The previous CNN based models were just to stack convolutional layers deeper to get better performance but in inceptions, wide architectures are introduced [28]. Computational cost is reduced with inception. For each layer, V1 uses 5x5 transformation of convolution and 3x3 convolution with max pooling. The results of these layers are concatenated in single output. 1x1 convolution is introduced which controls the depth. Global average pooling is used at the end of the network which take the average of a feature map and get a scalar value against each feature map. The parameters are reduced to 4 million. Softmax based two auxiliary classifiers are introduced to compute the performance within the network which helps to kept middle part alive. Auxiliary loss is used in training which is then added to total loss with a weight of 0.3. Version 2 introduces batch normalization which can help to avoid covariant shift [16] and version 3 introduces an architecture where factorization is involved which helps to minimize the multiplication cost [29].

3.3.4 ResNet

Residual Networks are considered as breakthrough which helps the development of deep networks. Researcher observed that adding more layers would have negative effect on performance of the network [14] because of the vanishing gradient problem. This problem occurred when while back propagating in deep network to earlier layers would make gradient very small because of repeated multiplications. ResNet introduced identity shortcut connection which skips layers and also known as residual block. We can use a skip connection to a traditional network to make if residual network.

3.3.5 Xception

This architecture introduces a depth wise separable convolution which basically minimizes the computational cost. It is interpretation of inception modules. It is the extreme version of inception [13]. Traditional convolutional is all about the correlation of spatial and depth but in this architecture the spatial correlation is mapped for each channel separately and then 1x1 convolution is performed for cross channel correlation. Hence there are two operation of convolutions are involved, first with channel wise convolution then point wise convolution with 1x1 convolutions.

3.3.6 DensNet

DensNet architecture introduces the idea of utilizing the features of the earlier layers. The DensNet architecture is similar to the ResNet architecture, which has produced better results than ResNet [15]. Dense blocks are used instead of Residual blocks in DensNet. Layers in DensNet are narrow which produce smaller number of features. Instead of summing the features, they are concatenated. And transition layer is introduced, which reduces the size of input. Transition layer has convolution with kernel size of 1 it is followed by 2x2 average pooling with stride equal to 2.

Chapter 4

Methodology

In this chapter we will discuss about the approach which is followed for classification purposes of malicious and benign images. There is couple of steps before directly training our deep learning based network. To make JPEGs malicious it is learnt that there segments are manipulated where scripts are concealed by the cyber-criminals. First those segments are converted to an image, which is then used in deep learning based approach.

4.1 Dataset

The dataset for our problem has been collected from CRC lab Bahria University. Files included in our dataset, are collected from honeypots installed at various locations. Dataset comprises of all JPEG files which are of JFIF format of JPEGs. There are various malicious files in the dataset which were used for malicious activity. They carry different scripts in their segments to achieve different objectives.

Dataset has 2200 files. Both classes has equal ratio of files in the dataset.There are 1100 files for Malicious class and 1100 for Benign class. Data preparation is implemented on the collected files, where script concealble segments are identified and converted to gray-scale images of fixed size of 24x24. Deep learning based approach can easily be implemented for classification of malicious and benign images.

4.2 Data Prepration

This phase comprised of two step, firs is identifying concealable segments and second is conversion into gray-scale images for further training process.

4.2.1 Segment Identification

It is learnt that JPEG' s JFIF format is widely used for malicious attacks [4, 2] and there are certain segments of images which are targeted by the cyber-criminals to carry the script. The structure of the JPEG is already discussed where the section 3.1 explains the segments in the JFIF format. The concealable part starts from very first marker known as start of image (SOI) marker, to the start of scan (SOS) marker. Segments between these markers are manipulated. After SOS marker the compressed data of image is present till the last marker which is end of image (EOI) marker.

4.2.2 Gray-Scale Conversion

Data between SOI and SOS is read in the hex from. To form the hex data into gray-scale image, hex vector $H = \{h1, h2, h3, \ldots, hn\}$ is converted to byte vector $B = \{b1, b2, b3, \ldots, bm\}$. The length of the data can have little variation because of different JPEGs and the scripts' length embedded in images. Figure 4.1 shows the steps of conversion the dataset into gray-scale images.

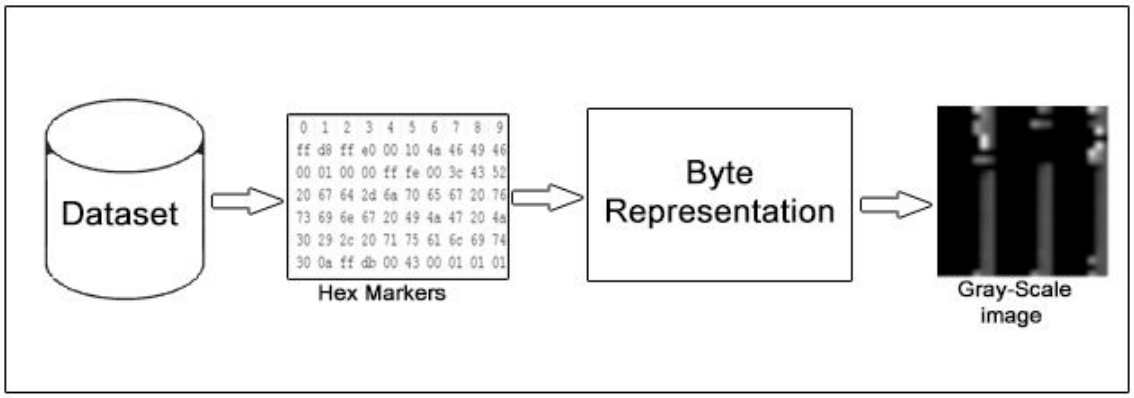

Figure 4.1: Conversion Process to gray-scale image

On visualizing the data it is learnt that malicious and benign images have little difference in their patterns of gray-scale images. Size was fixed to 24x24 dimensions for the images. For variation in the lengths of the segments data zero padding is added for the

additional pixels of the gray-scale to make it 24x24. Visual representation of the malicious and benign images are shown in figure4.2.

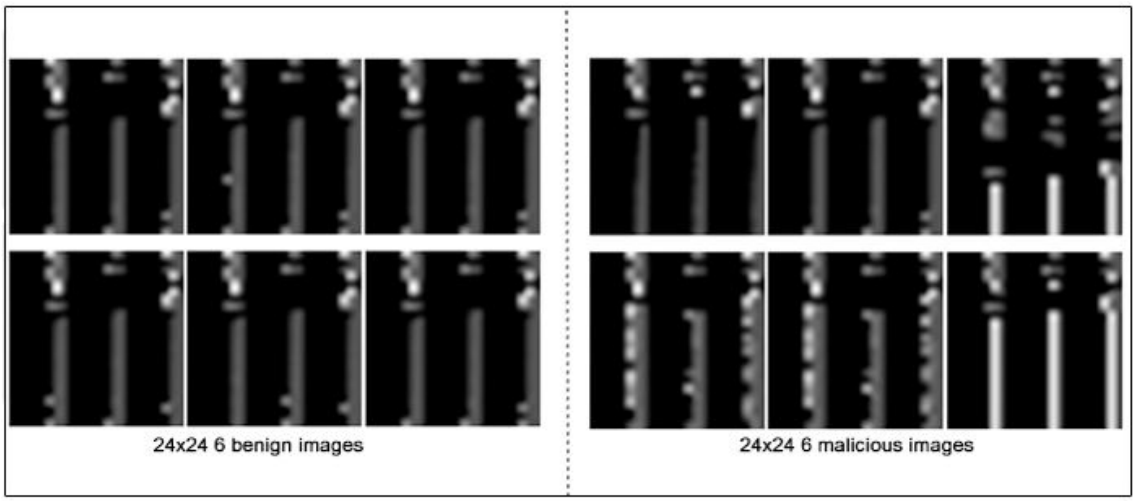

Figure 4.2: Benign and Malicious gray-scale images visulization

6 images from each class in above figure show that there are some pattern difference between the pixels of the malicious and the benign images. Using a deep learning based network can help to classify they benign or malicious images. We have employed Convolutional Neural Network (CNN) to extract features automatically and classify images with embedded script. In the following section overview of CNN is presented.

4.3 Convolutional Neural Network

This deep learning method is most popular method for extracting feature and employing classification based on extracted features. CNN has achieved a certain height specially regarding to the image classification problems. This method is different to the traditional classification problems where hand-crafted features are involved with some classifier e.g. Random Forest, Decision Tree, SVM etc. In CNN instead of hand-crafted features, automatically learnt features are used.

Features in CNN are learnt by passing the training data through multiple convolutional layers along with other layers. It is to note that the earlier layers in CNN model learn more general and high level features and as we go deep in to the network the learnt feature are more specific and more complex. CNN is basically comprises of two bases, one is feature extraction and other is classification.

A CNN comprises of many layers stacking up on each other. The layers in ConvNet have neurons arranged in 3 dimensions: width, height and depth. The layers in the CNN are briefly described below.

Input

This layer comprises of the input image which is fed to the Convolutional layer of neural network.

Convolutional Layer

Convolution in mathematics is merging of two sets of information. In this layer number of different filters convolve around the input image and produces the feature map in response to each filter in that layer. Features learnt could be squares or semicircles which are then fed to the next layer.

ReLU Layer

ReLU is an activation function like tanh and sigmoid. It is preferred on Tanh and Sigmoid because network trains very fast than the network with Tanh and Sigmoid activation functions without affecting the performance of the network. ReLU activation is applied on the output of the convolutional layer which results in all positive values. The ReLU changes all negative value to zero and positive values remain same as the input to ReLU activation. The equation is following where x is input to ReLU:

$$f(x) = max(0,x)$$

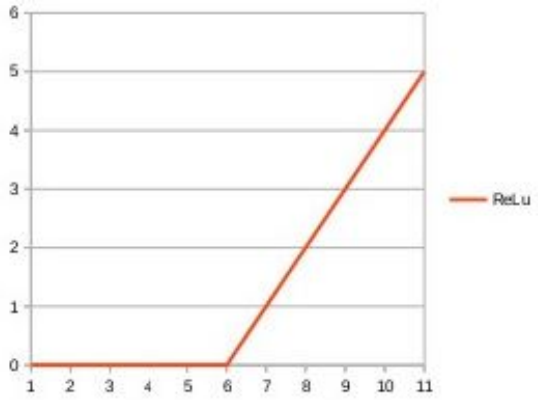

Figure 4.3: Graphical representation of ReLU activation

Pooling Layer

Pooling layer is applied on the feature map. Pooling basically serves two purposes, one is to reduces the parameters and second is to reduce the over fitting which results a better generalization. Pooling layer performs the down sampling on the activation map. Most important of Poolings are Max pooling and average pooling. In max pooling max value in receptive field is the output and in average pooling output is the average of all values in receptive filed.

Fully Connected Layer

This layer is usually stacked before the classifier is applied. This layer turns the feature maps to a 1D array. This basically flattens the feature maps. It keeps all the values of the feature maps to a 1D vector form. It is stacked after convolutional and pooling layers. Last FC layer has number of neurons equal to the number of classes. In binary classification if we apply Sigmoid, there would be one neuron in last layer suggesting the probability P of one class. And probability of other class is $1 - P$.

Putting all layers together to form a proper Neural Network for 2 class image classification problem is shown in figure 4.4.

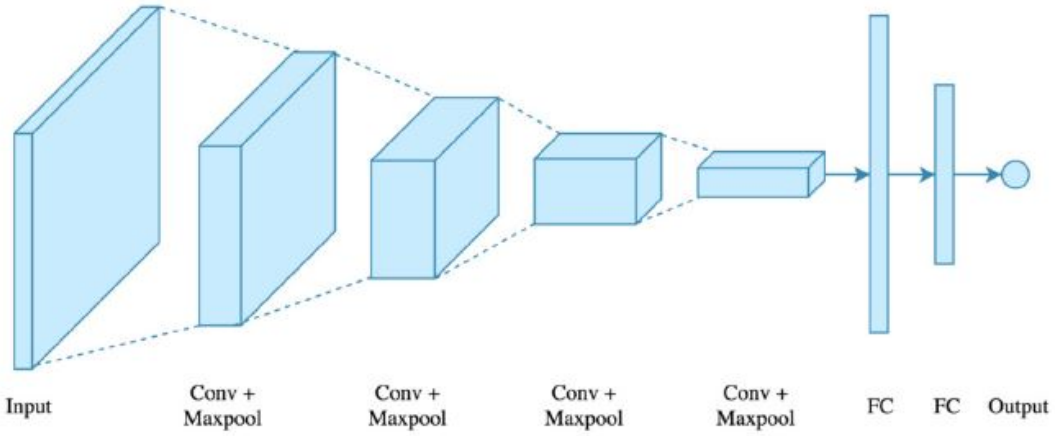

Figure 4.4: Architectural diagram of CNN

4.4 Model

The architecture of our CNN base approach is based on many convolutional layers, max pooling layers, dropout layers and dense layers. We have used 32 filters of size 9x9 in

the first convolutional layer with ReLU activation function and we used padding 1. Then we applied another convolutional layer with 32 filters of size 5x5 and used same activation function as above. We applied the max pooling layer with pooling size 2x2 with same stride. Further two convolutional layers are used with same number of hyper parameters which have 64 filters of size 3x3 and added padding 1 to one of these convolutional layer . Activations function is same as above convolutional layers. Then the activations of 4th convolutional layer is converted to vector form. After that fully connected layer of 64 neurons is implemented with a Dropout layer having dropout rate to be equal to 0.5. After it another dense layer of 32 neurons is used with ReLU activation function. Then a fully connected layer with 2 neurons with ReLU and at the end Sigmoid based classifier is used with 1 neuron due to binary classification problem. The structure of our model is present in figure 4.5.

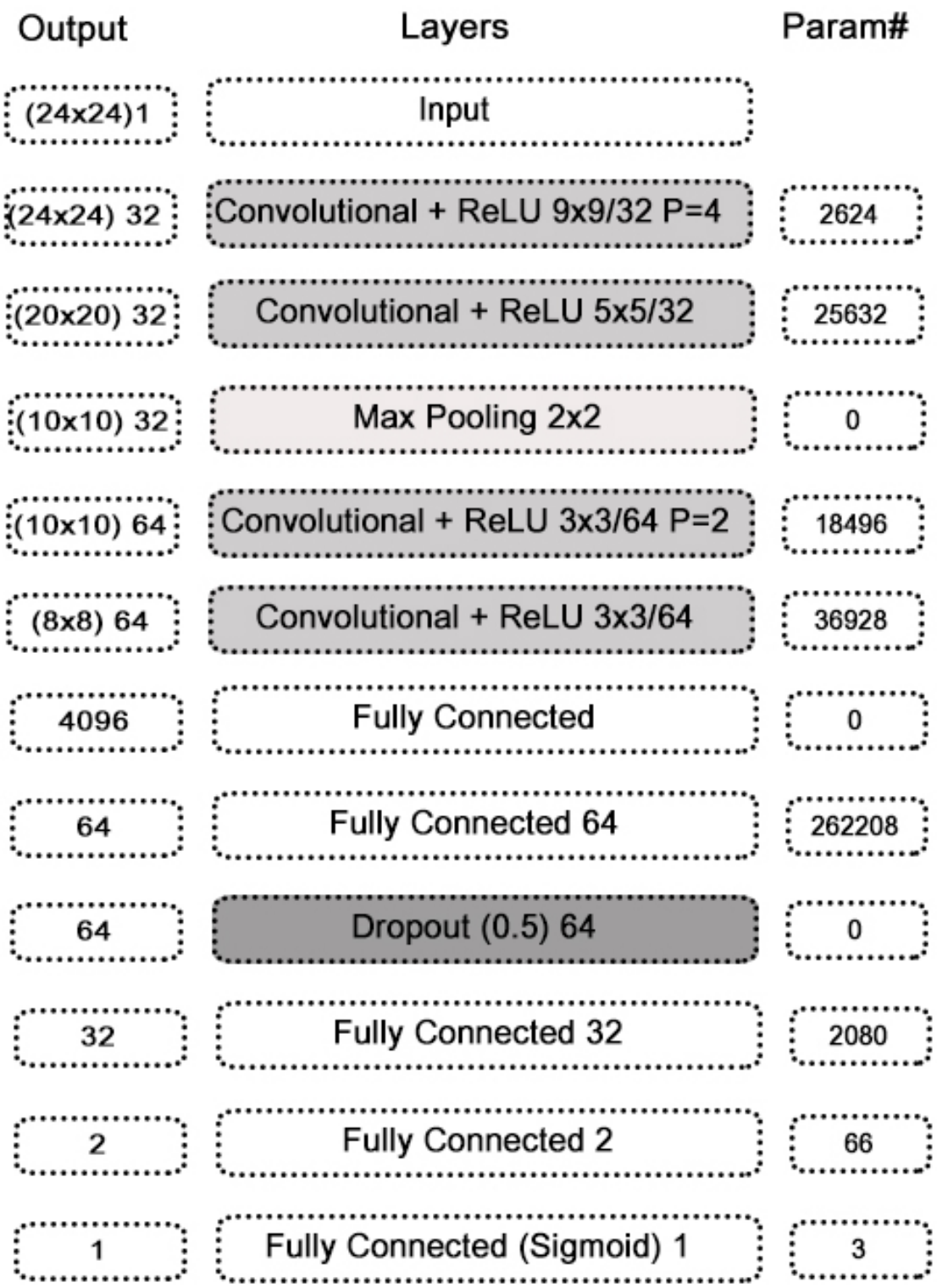

Figure 4.5: Our CNN based Model diagram

We have used same hyper parameters for all the architectures. 25 epochs are used

to train the networks with batch gradient descent. An epoch is a hyper-parameter, which controls the number of complete passes of training instances through the network. We used Batch Gradient Descent which means that all training examples are passed through the network before the model' s internal parameters are updated.

$$BatchSize = SizeofTrainingSet \tag{4.1}$$

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id} \tag{4.2}$$

where Δw represents weight to be updated, η is learning rate, t is target output, o is output and x represents training instances.

We are using Sigmoid based classifier for our binary classification problem, the loss function we are using is binary cross entropy. The mathematical representation of binary cross entropy/ log loss is following

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i)) \tag{4.3}$$

where N is total instances and y represents class labels (0 for Benign class and 1 for Malicious class).

Some pre-trained ConvNet architectures are used to evaluate the performance of our architecture. Detailed discussions of these architectures are included in section 3.3. Experiments and results based on these architectures are presented in next chapter.

Chapter 5

Experiments and Results

This chapter includes the results of the experiments carried out while employing deep learning based approach. The results are combined of various graphs and charts which depict the accuracies and the loss while training of our Convolutional Neural Network.

The results of the proposed method comprise of analysis of CNN with various parameters. We are feeding the network with the input of gray-scale images. The gray-scale images are of a fixed size which is discussed in Methodology. The dimensions are 24x24 for each image of both Benign and Malicious classes. The approach used in our experiments is well explained in Methodology.

5.1 Performance Measure

The performance of our proposed approach is computed against accuracies. We have split our data with the ratio of 3:7. 70% of our data is used for training and 30% of our data is used for the testing purposes. Our data is evenly distributed among both Benign and Malicious classes. For the summary of predicted results of our binary classification problem, we used Confusion matrix. Predictions with correct and incorrect class labels are summarized which includes the percent of count with each class. Precision and Recall are presented along confusion matrix. To measure the performance of binary classification we have used receiver operating characteristic (ROC) curve.

5.2 Experiment based on binary Classification using CNN

The dataset in gray-scale prepared from the header of malicious and benign images. The concealable segments of each JPEG are read in hex form. The concealable part starts from Start of Image SOI maker and ends at the Start of Scan marker. The values of hex vector are converted to byte format, which are then converted to gray-scale images, details in section 4.2.2. The gray-scale images are of 24x24. We applied CNN model to train on our dataset using training set and then compiled the results using test set. Our CNN model has achieved high accuracy of 96% with low loss which is equal to 0.14.

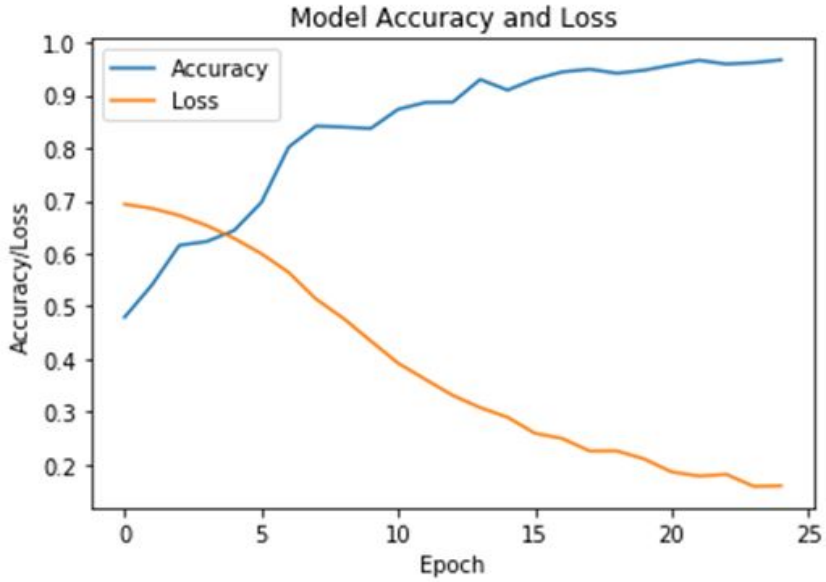

Figure 5.1: Graphical representation of training results

Figure 5.1 shows that with increasing the number of epochs it is learnt that the accuracy of our model during training increases. There is significant increase in the accuracy of our model with proportion to increase in epochs till 15 epochs. After 15 epochs the accuracy of our model gets minor increase with proportion to the increase in number of epochs. The loss as shown in above graph decreases constantly having significant change with increase of number of epochs. After 17 epochs, loss of our model starts to decrease constantly with a little lower rate till 25th epoch.

5.2.1 Evaluation and Credibility

We have plotted the confusion matrix for analyzing the predicted capability of our model. The accuracy on test data is 96% and the loss on test data is 0.14.

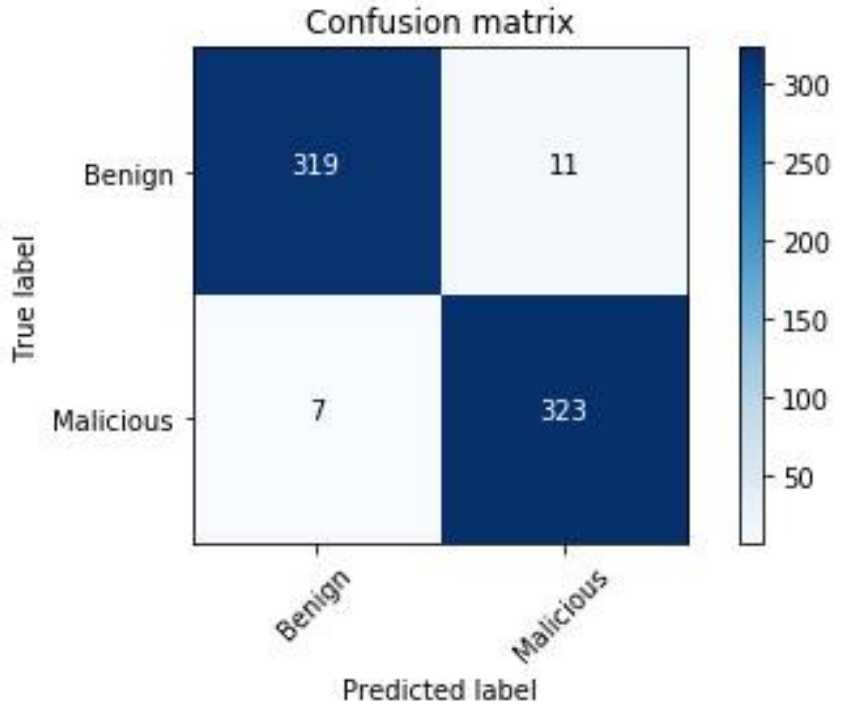

Figure 5.2: Confusion Matrix on test results

Confusion matrix on test results of binary calssification problem is presented in figure 5.2. It shows that 3% of images of Malicious class were classified as Benign image. And there is 4% of Benign images are predicted as Malicious image. The precision and recall of our model is presented below. The recall for benign and precision for malicious images is equal to 0.97. F-measure is same for both of the classes.

Class Label	**Precision**	**Recall**	**F-measure**	**Support**
Benign	0.96	0.97	0.97	337
Malicious	0.97	0.96	0.97	323

Table 5.1: Precision and Recall with other measures

ROC curve is at the following which shows our predicted results of binary classification problem. As the curve shows it has higher specificity (False Positive Rate) associated with higher sensitivity (True Positive Rate).

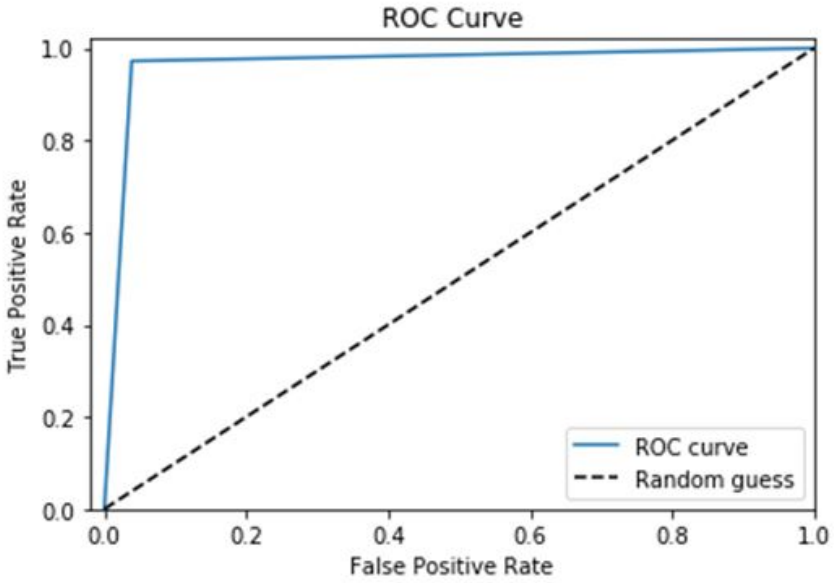

Figure 5.3: Graphical representation of ROC curve

5.3 Experiment on Pre-trained Models

We have used some pre-trained neural network. These networks are discussed in section 3.3. To fine tune these pre-trained models we prepared dataset to have 75x75 dimension for each image. The hyper parameters are similar to the parameters used in our CNN model.

The given chart shows the accuracy and the loss of these architectures.

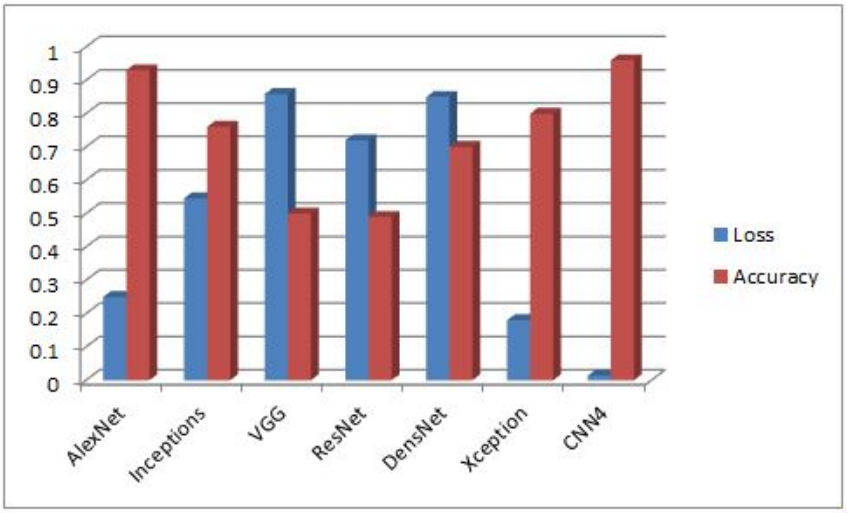

Figure 5.4: Loss and Accuracy of ConvNet Architectures on test data

It is learnt that the simpler model produces better results on our dataset. While training, the performances of ConvNet architectures were better but the results on test data of VGG, ResNet and Xception are not satisfactory. AlexNet performed way batter as compare to other pre trained models.

Training of the pre trained networks produces acceptable results. 25 epochs are used in training with batch gradient descent. The accuracy increases with increase in number of epochs and the loss decreases with increase in number of epochs.

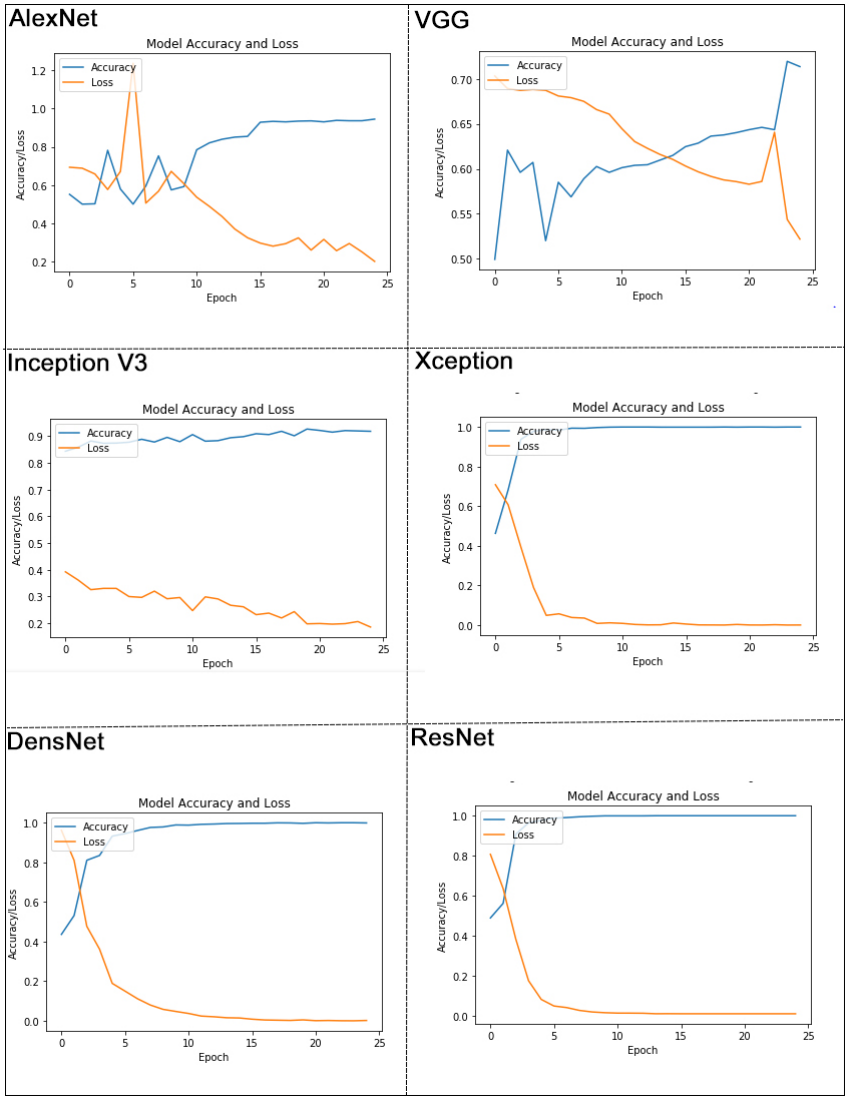

Figure 5.5: Loss and Accuracy of ConvNet Architectures while training

Chapter 6

Conclusion

In proposed study, the detection of malicious JPEGs is employed using a deep learning based technique. We have trained and well test Convolutional Neural Network.Segments which are concealable with malicious scripts, are segmented out by identifying different markers in JPEG structure. That JPEGs segments are in Hexadecimal form, which are then converted into gray-scale image. Experiments are carried out on those gray-scale images using CNN.

It was a binary classification problem where one class named Benign represents the harmless images with no manipulation and other class is named as Malicious which represented the JPEGs with malicious content. The cyber-criminal activities through JPEG images can be minimized which would assure the security of the systems, databases and some social network sites. Our proposed approach has achieved acceptable results for malicious JPEGs detection. The neural network has achieved 96% accuracy with minimum loss.

Further studies include, carrying the detection process to other image formats. The deep learning methods for the detection of those images can be employed with some mature classifiers. Further extension can be the extraction of the malicious script and analysis of that script, so that purpose of cyber-criminal activity can be identified. And other precautions related to security can be assured.

Bibliography

[1] Hiding in plain sight: a story about a sneaky banking trojan. https://blog.malwarebytes.com/threat-analysis/2014/02/hiding-in-plain-sight-a-story-about-a-sneaky-banking-trojan/. Accessed: 16th Oct,2018. Cited on pp. 7 and 13.

[2] The little jpeg that could (hack your organization). https://www.rsaconference.com/events/us15/agenda/sessions/1513/the-little-jpeg-that-could-hack-your-organization. Accessed: 16th Oct,2018. Cited on pp. 2, 3, 6, 13, and 23.

[3] The metadata in jpeg files. https://dev.exiv2.org/projects/exiv2/wiki/The_Metadata_in_JPEG_files. Accessed: 30th Nov,2018. Cited on p. 11.

[4] Syscan'15 singapore: Stegosploit - hacking with pictures. https://www.youtube.com/watch?v=np0mPy-EHII. Accessed: 14th Oct,2018. Cited on pp. 2, 3, 6, and 23.

[5] Wiki: Ascii. https://en.wikipedia.org/wiki/ASCII. Accessed: 30th April,2019. Cited on p. 17.

[6] Wiki: Ascii. https://en.wikipedia.org/wiki/Extended_ASCII. Accessed: 30th April,2019. Cited on p. 18.

[7] Wiki: Base64. https://en.wikipedia.org/wiki/Base64. Accessed: 25th April,2019. Cited on pp. 16 and 17.

[8] Wiki: Binary number. https://en.wikipedia.org/wiki/Binary_number. Accessed: 5th April,2019. Cited on p. 14.

[9] Wiki: Hexadecimal. https://en.wikipedia.org/wiki/Hexadecimal. Accessed: 6th April,2019. Cited on p. 14.

[10] Wiki: Jpeg. https://en.wikipedia.org/wiki/JPEG. Accessed: 30th Nov,2018. Cited on pp. 12 and 13.

[11] Wiki: Octal. https://en.wikipedia.org/wiki/Octal. Accessed: 7th April,2019. Cited on p. 14.

[12] Shikha Badhani and Sunil K Muttoo. Evading android anti-malware by hiding malicious application inside images. *International Journal of System Assurance Engineering and Management*, pages 1–12, 2018. Cited on p. 8.

[13] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. Cited on p. 20.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. Cited on p. 20.

[15] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. Cited on p. 21.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. Cited on p. 20.

[17] Blossom Kaur, Amandeep Kaur, and Jasdeep Singh. Steganographic approach for hiding image in dct domain. *International Journal of Advances in Engineering & Technology*, 1(3):72, 2011. Cited on p. 9.

[18] Dong-Hyun Kim and Hae-Yeoun Lee. Deep learning-based steganalysis against spatial domain steganography. In *2017 European Conference on Electrical Engineering and Computer Science (EECS)*, pages 1–4. IEEE, 2017. Cited on p. 8.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. Cited on p. 19.

[20] Ajit Kumar, K Pramod Sagar, KS Kuppusamy, and G Aghila. Machine learning based malware classification for android applications using multimodal image representations. In *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, pages 1–6. IEEE, 2016. Cited on p. 6.

[21] Rakesh Singh Kunwar and Priyanka Sharma. Framework to detect malicious codes embedded with jpeg images over social networking sites. In *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pages 1–4. IEEE, 2017. Cited on pp. 1 and 7.

[22] Shahid Latif, Rahat Ullah, and Hamid Jan. A step towards an easy interconversion of various number systems. *arXiv preprint arXiv:1107.1663*, 2011. Cited on pp. 13 and 14.

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. Cited on p. 19.

[24] Ching-Yung Lin and Shih-Fu Chang. A robust image authentication method distinguishing jpeg compression from malicious manipulation. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(2):153–168, 2001. Cited on p. 9.

[25] Liu Liu, Bao-sheng Wang, Bo Yu, and Qiu-xi Zhong. Automatic malware classification and new malware detection using machine learning. *Frontiers of Information Technology & Electronic Engineering*, 18(9):1336–1347, 2017. Cited on p. 5.

[26] Andysah Putera Utama Siahaan. Base64 character encoding and decoding modeling. Cited on pp. 15 and 16.

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Cited on p. 19.

[28] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. Cited on p. 20.

[29] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. Cited on p. 20.

[30] Jian Ye, Jiangqun Ni, and Yang Yi. Deep learning hierarchical representations for image steganalysis. *IEEE Transactions on Information Forensics and Security*, 12(11):2545–2557, 2017. Cited on p. 8.

Malicious JPEG detection using Deep Learning

ORIGINALITY REPORT

8%	4%	2%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Higher Education Commission Pakistan Student Paper	1%
2	Submitted to University of Bristol Student Paper	<1%
3	Submitted to University of Warwick Student Paper	<1%
4	dev.exiv2.org Internet Source	<1%
5	Submitted to National University of Singapore Student Paper	<1%
6	www.misp-project.org Internet Source	<1%
7	Submitted to Marist College Student Paper	<1%
8	www.cse.cuhk.edu.hk Internet Source	<1%
9	Submitted to The University of Manchester	

	Student Paper	<1%
10	Aws Anaz, Marjorie Skubic, Jay Bridgeman, David M. Brogan. "Classification of Therapeutic Hand Poses Using Convolutional Neural Networks", 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2018 Publication	<1%
11	Submitted to University of Sunderland Student Paper	<1%
12	"Cognitive Informatics and Soft Computing", Springer Science and Business Media LLC, 2019 Publication	<1%
13	Submitted to American Public University System Student Paper	<1%
14	Submitted to University of Queensland Student Paper	<1%
15	www.cweldtech.com Internet Source	<1%
16	Jian Ye, Jiangqun Ni, Yang Yi. "Deep Learning Hierarchical Representations for Image Steganalysis", IEEE Transactions on Information Forensics and Security, 2017	<1%

	Publication	
17	"Natural Language Processing and Information Systems", Springer Nature, 2018 Publication	<1%
18	Submitted to iGroup Student Paper	<1%
19	Submitted to Napier University Student Paper	<1%
20	www.i-scholar.in Internet Source	<1%
21	kiforok.blogspot.com Internet Source	<1%
22	Ahmad, Asma, G.R. Sinha, and Nikita Kashyap. "3-Level DWT Image Watermarking Against Frequency and Geometrical Attacks", International Journal of Computer Network and Information Security, 2014. Publication	<1%
23	cacm.acm.org Internet Source	<1%
24	nigelborrington.ie Internet Source	<1%
25	myaps.blogspot.com Internet Source	<1%

26	media.proquest.com Internet Source	<1%
27	www.lrwh.us Internet Source	<1%
28	Submitted to Uxbridge College, Middlesex Student Paper	<1%
29	www.springerprofessional.de Internet Source	<1%
30	Submitted to Rochester Institute of Technology Student Paper	<1%
31	www.rsaconference.com Internet Source	<1%
32	es.scribd.com Internet Source	<1%
33	Submitted to New Jersey Institute of Technology Student Paper	<1%
34	"Grid and Cloud Database Management", Springer Nature, 2011 Publication	<1%
35	scholarbank.nus.edu.sg Internet Source	<1%
36	archive.org Internet Source	<1%

37	Sartid Vongpradhip, Suppat Rungraungsilp. "QR code using invisible watermarking in frequency domain", 2011 Ninth International Conference on ICT and Knowledge Engineering, 2012 Publication	<1%
38	Xiaoyan Xu, Linna Fan, Rongcai Zhao. "Steganalysis of Content-Adaptive JPEG Steganography Based on CNN and 2D Gabor Filters", Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence - CSAI '18, 2018 Publication	<1%
39	dspace.thapar.edu:8080 Internet Source	<1%
40	Submitted to University of Glamorgan Student Paper	<1%
41	Submitted to Universidad Nacional de Colombia Student Paper	<1%
42	Konstantinos Karampidis, Ergina Kavallieratou, Giorgos Papadourakis. "A review of image steganalysis techniques for digital forensics", Journal of Information Security and Applications, 2018 Publication	<1%

43	Submitted to Champlain College Student Paper	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		