

MALICIOUS WEB TRAFFIC DETECTION USING SPATIAL PYRAMID POOLING WITH DEEP LEARNING



By

Abdur Rashid

MS-IS-330628

Supervisor

Dr. Hasan Tahir

Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science in Information Security (MS IS),

In

School of Electrical Engineering and Computer Science,


National University of Sciences and Technology (NUST),

Islamabad, Pakistan.

(March 2024)

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis entitled "Malicious Web Traffic Detection Using Spatial Pyramid Pooling with Deep Learning" written by Abdur Rashid, (Registration No 00000330628), of SEecs has been vetted by the undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

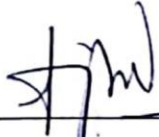
Signature: _____ 

Name of Advisor: Dr. Hasan Tahir

Date: 13-Feb-2024

HoD/Associate Dean: _____ 

Date: 20-02-2024

Signature (Dean/Principal): _____  **Dr. Muhammad Ajmal**
Principal
NUST School of Electrical
Engg & Computer Science
H-12, Islamabad

Date: 20 Feb, 2024

Approval

It is certified that the contents and form of the thesis entitled "Malicious Web Traffic Detection Using Spatial Pyramid Pooling with Deep Learning" submitted by Abdur Rashid have been found satisfactory for the requirement of the degree

Advisor : Dr. Hasan Tahir

Signature:  _____

Date: 13-Feb-2024

Committee Member 1:Dr. Sana Qadir

Signature:  _____

13-Feb-2024

Committee Member 2:Dr. Mehdi Hussain

Signature:  _____

Date: 13-Feb-2024

Signature:  _____

Date: 14 Feb 2024

Dedication

In grateful appreciation of the unwavering love, guidance, and encouragement bestowed upon me by my dear parents, and the constant support and camaraderie of my siblings, particularly my brother, whose unwavering assistance has lit up my journey.

Certificate of Originality

I hereby declare that this submission titled "Malicious Web Traffic Detection Using Spatial Pyramid Pooling with Deep Learning" is my own work. To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics, which has been acknowledged. I also verified the originality of contents through plagiarism software.

Student Name: Abdur Rashid

Student Signature: 

FORM TH-4

National University of Sciences & Technology

MASTER THESIS WORK

We hereby recommend that the dissertation prepared under our supervision by: (Student Name & Reg. #) Abdur Rashid [00000330628]

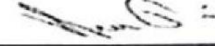
Titled: Malicious Web Traffic Detection Using Spatial Pyramid Pooling with Deep Learning

be accepted in partial fulfillment of the requirements for the award of Master of Science (Information Security) degree.

Examination Committee Members

1. Name: Sana Qadir Signature: 
29-Feb-2024 12:33 PM

2. Name: Mehdi Hussain Signature: 
29-Feb-2024 12:33 PM

Supervisor's name: Hasan Tahir Signature: 
29-Feb-2024 12:39 PM



HoD/Associate Dean

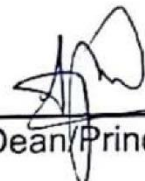
01-03-2024

Date

COUNTERSIGNED

01 MAR 2024

Date


Dr. Muhammad Ajmal
Principal
NUST School of Electrical
Engg & Computer Science
NUST, Islamabad

Dean/Principal

Acknowledgment

First, I am thankful to ALLAH (SWT) for all the good things in my life.

I am grateful to my research supervisor, Dr. Hasan Tahir, for his valuable guidance and unwavering support during my research, especially during times when I needed direction.

I extend my deepest gratitude to my parents for their endless love, prayers, sacrifices, and motivation for my bright future. I am also thankful to my siblings for their unwavering support and invaluable prayers.

TABLE OF CONTENTS

Dedication	i
Acknowledgment	iv
TABLE OF CONTENTS	vii
LIST OF ABBREVIATIONS	x
LIST OF TABLES	xi
LIST OF FIGURES	xii
ABSTRACT	xiii
INTRODUCTION	1
1.0 Background.....	1
1.1 Problem Statement	2
1.2 Research Motivation.....	3
1.3 Web Attacks	4
1.4 Convolutional Neural Networks (CNNs) in Deep Learning.....	7
1.4.1 Introduction to CNNs.....	7
1.4.2 Architectural Overview.....	8
1.4.3 Functionality and Application.....	8
1.5 Spatial Pyramid Pooling	9
1.5.1 Input Image	10
1.5.2 Feature Maps of conv5 (Arbitrary Size)	10
1.5.3 Spatial Pyramid Pooling Layer	10
1.5.4 Fully Connected Layers (fc6, fc7)	11
1.6 Research Objectives	11
1.7 Thesis Organization.....	12
1.8 Thesis Contributions.....	12
1.8.1 Development of Deep Learning Model	12
1.8.2 Curated Dataset of Web Attack Types.....	12
1.8.3 Empirical Model Evaluation	13

1.9	Summary.....	13
LITERATURE REVIEW.....		14
2.0	Threat Landscape	14
2.0.1	Industries Targeted by Cyber Attacks.....	15
2.0.2	Top Attacking Countries.....	16
2.0.3	Top Violation Types	17
2.1	Existing Approaches in Web Security	18
2.1.1	Application Level Mitigations	18
2.1.2	Machine Learning Methods	20
2.1.3	Deep Learning Methods.....	21
2.2	Summary.....	25
RESEARCH METHODOLOGY		26
3.0	Research Methodology Workflow	26
3.1	Experimental Design and Procedures	27
3.1.1	Deployment of Web Applications in AWS Cloud.....	27
3.1.2	Planning and Architectural Design	27
3.1.2.1	AWS Account and Security Configuration	28
3.1.2.2	Service Selection	28
3.1.2.3	Application Code Deployment	28
3.1.2.4	Database and Storage.....	28
3.1.2.5	Load Balancing and Auto-scaling	28
3.1.2.6	Compliance with AWS Best Practices	29
3.1.3	Data Ingestion	30
3.1.4	Data Labeling.....	32
3.1.5	Data Conversion.....	35
3.1.6	CNN Training	37
3.1.6.1	Model Architecture.....	37
3.1.6.2	Optimization and Compilation	39
3.1.6.3	Training Process	39
3.2	Experimental Network Diagram	39
3.3	Proposed Network Design.....	41
3.4	Device Specifications	42
3.5	Summary.....	42

EXAMINATION AND ANALYSIS	43
4.0 Evaluation Metrics and Results Presentation	43
4.0.1 Accuracy	43
4.0.2 Precision.....	43
4.0.3 Recall	44
4.0.4 F1 Score	44
4.0.5 Support.....	45
4.0.6 Macro Average.....	45
4.0.7 Weighted Average	45
4.1 Confusion Matrix Analysis	46
4.1.1 Metrics Calculation Formulae.....	48
4.1.2 Accuracy and Loss Metrics across Epochs	49
4.2 Comparative Analysis	50
4.2.1 Categories-Based Accuracy	51
4.3 Summary.....	51
CONCLUSION AND FUTURE HORIZONS.....	53
5.0 Conclusion.....	53
5.0.1 Overview.....	53
5.0.2 Key Findings.....	53
5.0.3 Significance.....	54
5.0.4 Implications for Cyber Security.....	54
5.1 Future Research Directions	54
REFERENCES	55

LIST OF ABBREVIATIONS

CNN - Convolution Neural Network

SPP - Spatial Pyramid Pooling

CTI - Cyber Threat Intelligence

IDS - Intrusion Detection System

IPS - Intrusion Prevention Systems

SIEM - Security Information and Event Management

OWASP - Open Worldwide Application Security Project

XSS - Cross-Site Scripting

SSRF - Services Side Request Forgery

SQLI - SQL Injection

RCE - Remote Code Execution

LFI - Local File Inclusion

LFD - Local File Disclosure

XEE - XML External Entity

TTPs - Tactics, Techniques, and Procedures

ELB - Elastic Load Balancer

LIST OF TABLES

Table 3.1 Dataset Composition	35
Table 3.2 Specification of the Computing Device	42
Table 4.1 Performance Indicators	45
Table 4.2 Comparative Analysis.....	51
Table 4.3 Categories-Based Accuracy	51

LIST OF FIGURES

Figure 1.1 Number of Attacks by Attack Vectors	03
Figure 1.2 XSS Attack	04
Figure 1.3 SSRF Attack	05
Figure 1.4 SQLI Attack	06
Figure 1.5 Convolution Neural Network	09
Figure 1.6 Spatial Pyramid Pooling Layer Operation.....	10
Figure 2.1 Trend in Web Application Attacks from 2020 to 2022	15
Figure 2.2 Top Attacked Industries	16
Figure 2.3 Top Attacking Countries	17
Figure 2.4 Top Violation Types	18
Figure 3.1 Logs Ingestion Connector	31
Figure 3.2 Kusto Query Language in Azure Sentinel for Malicious Request Collection	32
Figure 3.3 String Search in KQL.....	33
Figure 3.4 Collected Payloads in CSV File	34
Figure 3.5 Greyscale Images	36
Figure 3.6 Python Code for Text to Image Conversion.....	36
Figure 3.7 CNN Model Architecture	38
Figure 3.8 Experimental Network Diagram.....	40
Figure 3.9 Proposed Network Diagram	41
Figure 4.1 Confusion Metrix on Test Dataset	47
Figure 4.2 Metrics Calculation Formulas	48
Figure 4.3 Model Accuracy and Loss Curves.....	50

ABSTRACT

The widespread use of the internet has brought immense convenience, but it has also led to a rise in cyber crimes. Attackers are using various tactics and techniques to compromise the security of information systems. One of the major threats in this landscape is web attacks, which pose a serious threat to web applications. Extensive work has been done for web security through multiple detection and prevention tools at each layer of security. Tools like IDS, IPS, and SIEM solutions have been proposed to detect and prevent these attacks. These security solutions mainly rely on network traffic stats (flows), signatures, cyber threat intelligence (CTI), and static threat detection rules. These methods have protected web security, but there are some limitations observed toward advanced attack payloads that use sophisticated techniques, a limited number of attempts, and zero-day exploits. This research aims to identify malicious web traffic using an innovative approach that combines deep learning with spatial pyramid pooling (SPP) to detect attacks on the base of payloads in network traffic. Deep learning is a powerful tool for recognizing patterns and extracting features from images. The proposed method involves using image classification techniques to dynamically spot different types of web attacks on the fly. By converting both malicious and clean payloads into image formats, the model has been trained to classify these data into either malicious or clean categories. Additionally, SPP techniques have been used to adapt the model to varying sizes of images. This method will help to improve the efficiency of the model by avoiding information loss due to resizing and cropping images to a fixed size. This work automates the process of extracting meaningful features, eliminating the need for manual feature selection commonly used in traditional machine learning approaches. The proposed approach aims to provide a more effective defense against evolving web attacks.

INTRODUCTION

1.0 Background

Web application plays a vital role in every aspect of life. Each day, many new web applications come into existence. According to the research in [1] the average person spends 6 hours and 58 minutes on the internet. Businesses, educational organizations, financial institutions, and health care organizations, in short, every government and private entity heavily relies on web applications to serve their customers. Therefore, this is the top priority to provide a secure cyber space to all users and to ensure their privacy.

OWASP TOP 10 [2] is a well-known and widely accepted report, which lists the most critical web application vulnerabilities. The OWASP Top 10 guides developers, security professionals, and organizations to help them prioritize and address common security vulnerabilities and risks in web applications. These vulnerabilities can lead to several types of attacks, including data breaches, unauthorized access, and more.

To ensure the security of web applications, several efforts have been made to be safe from these types of attacks on the application level. Techniques like checking and cleaning up the information users enter, secure coding, making sure that only trusted sources can access the apps, and even disabling certain features like JavaScript in browser that can be exploited by

attackers. These steps are essential, but the best defense is to prevent these attacks before they hit web applications.

1.1 Problem Statement

There are multiple solutions available like IDS, IPS, WAF, and SIEM to detect attacks against web applications. Sometimes new and advanced web attacks evade these solutions. The issue with these solutions is their static detection capabilities as these are mostly based on signature, strings, or pattern match and some static rules in SIEM solutions that correlate multiple events, but they are prone to new and more advanced techniques. Researchers addressed this issue using machine learning based on network flows. In computer networks, a "flow" is a group of data packets that are similar in some ways and treated as one unit. These packets have certain characteristics in common like source, destination, protocols, port number, etc. that help to see how much data is moving through the network and spot any unusual activity or security risks. Tools like NetFlow, sFlow, and IPFIX are used in networks to collect information about these flows [3]. However flow based detection methods, face challenges in identifying advanced web attacks like SQLI, XSS, SSRF, XEE, LFI, etc. These attacks often involve a small number of network flows but carry harmful content in the packet payload. Machine learning techniques require manual feature extraction for training which is time consuming for such a diverse range of payloads. Flow based detection is also susceptible to evasion techniques, like splitting the attack into smaller packets or spreading it across multiple hosts or IP addresses. The proposed solution involves a deep learning method utilizing Convolutional Neural Networks (CNNs) to thoroughly examine web request payloads and identify sophisticated web attacks.

1.2 Research Motivation

The internet serves as a fundamental pillar of the global infrastructure. An increasing number of online services are now available but keeping them secure from cyber threats is a constant challenge.

Attackers are using several ways to break into web applications, trying to get access to important data that they should not have. As organizations focus more on using technology to make things better and easier, the attackers are also working hard to find ways to enter online systems and steal important information.

The Akamai Threat Report 2022 [4] shows a significant increase in these malicious activities targeting web applications as shown in Fig 1.1. This increase shows for local file inclusion, SQL injection and remote file inclusion and other sophisticated attacks, where the adversaries intent on disruption and data theft shown in Fig 1.1 from 2021 to April 2022. These stats show the need for an adaptable and intelligent solution to detect and prevent these attacks. It pushes experts to find new and better ways to protect against these kinds of cyber attacks.

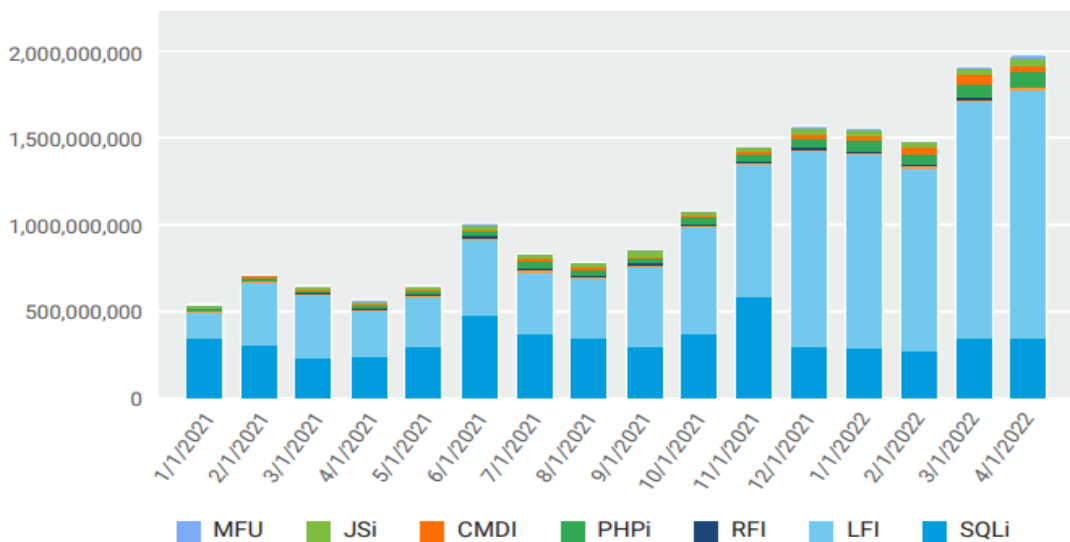


Figure 1.1 Number of Attacks by Attack Vectors [4]

The MITRE ATT&CK framework covers most of these tactics techniques and procedures (TTPs) in a logical order. The most common types of web attacks are cross sides scripting (XSS), services side request forgery (SSRF), SQL injection (SQLI), command Injection, remote code execution (RCE), local file inclusion (LFI), directory traversal, open redirection etc.

1.3 Web Attacks

Understanding web attack techniques is important for developing robust defense mechanisms and implementation of security measures that mitigate the risk of exploitation and protect against the potential. The following attacks have been addressed in this research work:

- **Cross-Site Scripting (XSS)** - this attack involves injecting malicious scripts into web pages viewed by other users. Cyber criminals exploit vulnerabilities in a legit website's code to execute scripts in the victim's browsers. These scripts can steal sensitive data, manipulate website content, or redirect users to malicious sites as shown in Fig 1.2.

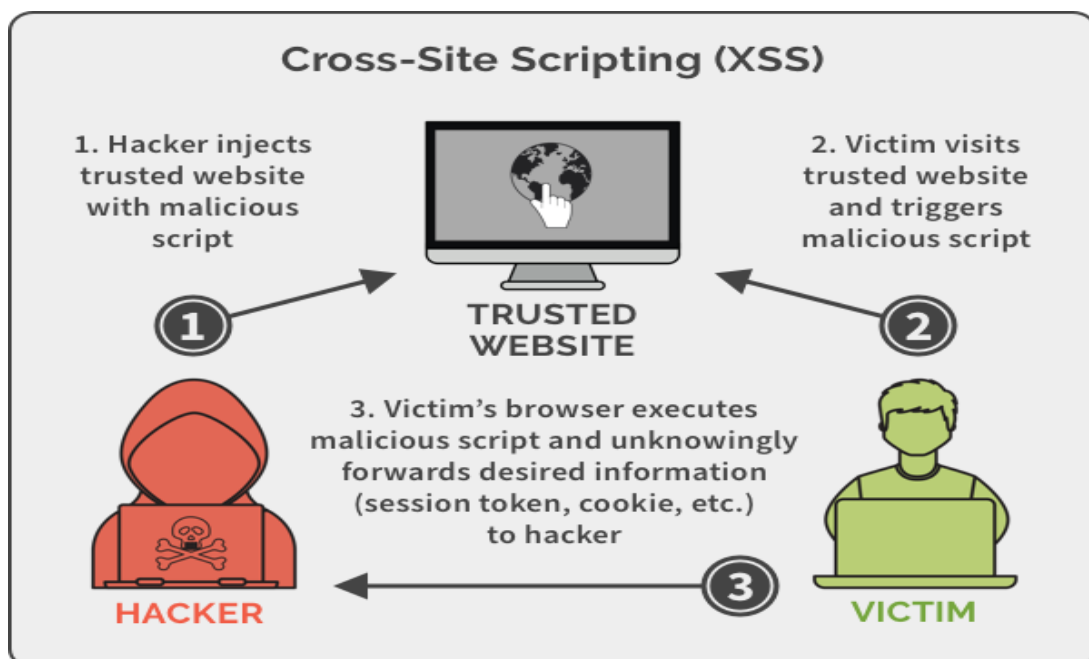


Figure 1.2 XSS Attack [5]

- **Server-Side Request Forgery (SSRF)** - SSRF occurs when attackers trick a logged-on user to click on manipulated link sent by the attacker into making requests on their behalf. By exploiting this vulnerability, attackers can steal private information and access internal systems, or perform actions on the server. Fig 1.3 shown the common attack vector for SSRF attack.



Figure 1.3 SSRF Attack [6]

- **SQL Injection (SQLI)** – this class of attack targets databases by injecting malicious SQL code through the input fields of a website. It can allow attackers to view, modify, or delete data from the database, potentially causing severe damage to the system's integrity [7]. In Fig 1.4 the attacker injecting an always true ($1=1$) string with an OR operator, will retrieve all rows from the teachers table in database.

SQL Injection

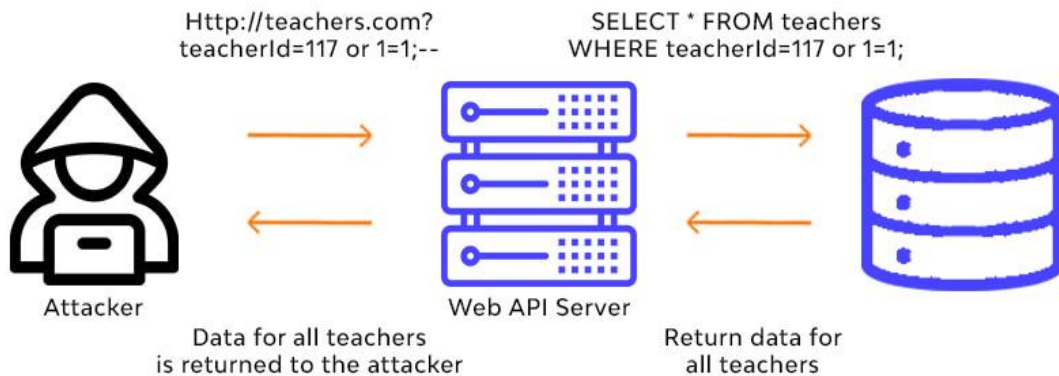


Figure 1.4 SQLI Attack [7]

- **Command Injection** - attackers exploit vulnerable applications by inserting malicious commands that enable them to execute arbitrary commands on the server, potentially gaining unauthorized access or manipulating system configuration [8].
- **Remote Code Execution (RCE)** - RCE attacks allow attackers to execute arbitrary code on a targeted system or server. This access can lead to taking control of the system, installing malware, modifying data, or performing other malicious activities [9].
- **Local File Inclusion (LFI)** - LFI attacks exploit code vulnerabilities to include files from the server into web pages. Attackers can use this to read sensitive files or execute arbitrary code, potentially compromising the entire system's security [10].
- **Local File Disclosure (LFD)** - LFD is a security vulnerability where an attacker gains unauthorized access to read sensitive files stored on a system or server. Instead of executing files, the focus is on accessing and viewing files that should be restricted from general access [11]. This vulnerability could expose critical information such as configuration details, user credentials, or other confidential data.

- **Directory Traversal** - this attack exploits insufficient input validation, allowing attackers to access files and directories outside the intended scope. It permits unauthorized access to critical files or resources on the server [12].
- **Open Redirection** - attackers manipulate URLs in vulnerable web applications to redirect users to malicious websites. They exploit these flaws to direct users to phishing sites or malware-infested pages, potentially compromising user security [13]. It is recommended to request users to submit a concise identifier, such as a brief name, ID, or token. This identifier should be linked on the server side to the complete target URL. This approach minimizes the risk of manipulation by ensuring that the mapping to the full URL is securely handled by the server, rather than relying on user-provided [14].
- **XML External Entity (XEE)** – this is a security loophole in web applications. This vulnerability enables an attacker to manipulate how the application deals with XML data. By exploiting this weakness, the attacker can access files stored on the application's server and potentially interact with other systems that the application [15].
- **Insecure Deserialization** - insecure deserialization occurs when a website processes user-controlled data that undergoes deserialization. This allows potential manipulation of serialized objects, allowing attackers to introduce harmful data into the application's code [16].

1.4 Convolutional Neural Networks (CNNs) in Deep Learning

1.4.1 Introduction to CNNs

Convolutional Neural Networks (CNNs) are the heart of deep learning, particularly within the domain of image recognition. These networks are specifically designed to process data in a grid-like topology, such as images, which makes them perfect for tasks involving visual inputs.

1.4.2 Architectural Overview

A typical CNN architecture is composed of several layers, each designed to perform specific operations on the input data. The key layers include convolutional layers, pooling layers, and fully connected layers [17]. Convolutional layers apply a series of filters to the input to create feature maps, capturing spatial hierarchies and patterns. Pooling layers, often following convolutional layers, reduce the spatial size of the representation, thus decreasing the computational load and mitigating overfitting. Fully connected layers, resembling traditional neural network layers, are used toward the end of the network for classification or regression tasks.

1.4.3 Functionality and Application

The strength of CNNs lies in their ability to automatically and adaptively learn spatial information from input images. This feature is crucial in tasks such as object detection, face recognition, and medical image analysis etc. Furthermore, CNNs are instrumental in other areas beyond image processing, including natural language processing and time series analysis, where the underlying principles of spatial feature learning can be applied. Recent advancements in CNNs involve the development of deeper and more complex architectures, such as ResNet, etc. which have significantly improved performance in various tasks [18]. The operation of a typical CNN layer include the convolution operation of input image and fixed size filters, which swap across all the pixels of the input image. Following the CNN layer there are max pooling layers that perform the subsampling operation (max pooling or average pooling) on the feature maps to reduce the number of features. In a traditional CNN model Fig 1.5 there are several CNN and max pooling layers in a sequence. The last layer contains the fully connected layer that perform the classification operation on the features extracted via CNN layers.

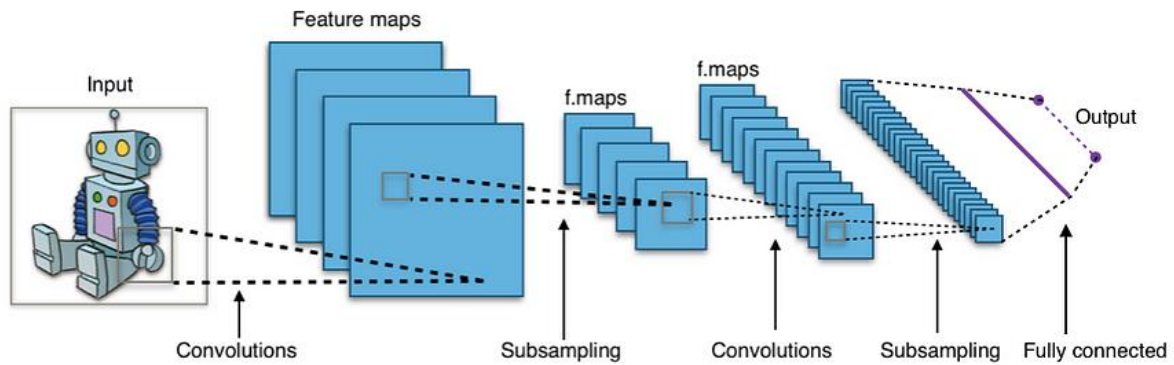


Figure 1.5 Convolution Neural Network [18]

1.5 Spatial Pyramid Pooling

Spatial Pyramid Pooling (SPP) is an innovative mechanism in deep learning, particularly within the architecture of convolutional neural networks (CNNs). The utility of SPP originated from its capability to maintain the structural integrity of input data by accommodating feature maps of variable sizes input images.

The operational principle of SPP lies in its hierarchical partitioning of the input feature map into segments, each pooling features independently. SPP ensures that features at various scales are encapsulated, offering a comprehensive representation of the input data. This is important because it preserves spatial relationships and contextual information, which are important for the accurate interpretation of complex data patterns.

Another advantage of SPP is its versatility. It can be integrated into existing CNN architectures, enhancing their ability to process non-uniform inputs without the need for distortion or resizing. This is different from traditional methods that often require a pre-processing step to achieve a uniform input size, potentially leading to loss of information and a reduction in model performance.

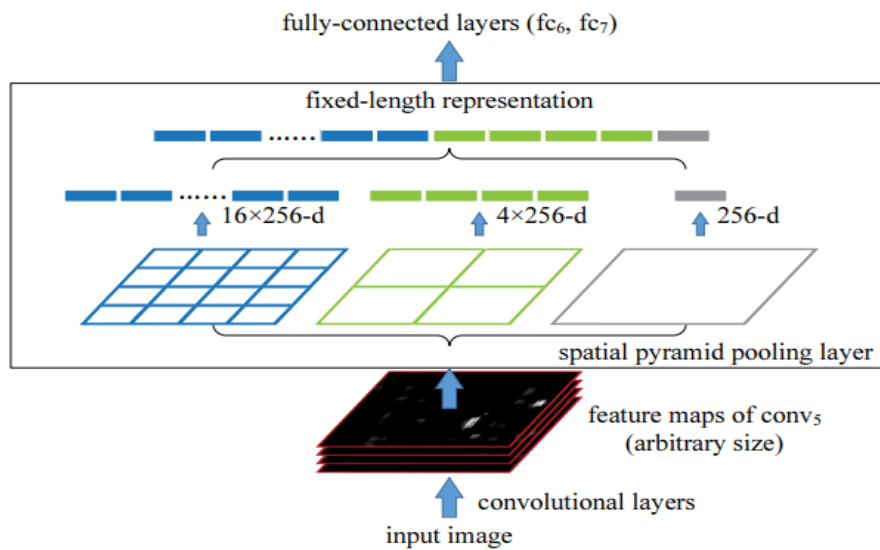


Figure 1.6 Spatial Pyramid Pooling Layer Operation [19]

1.5.1 Input Image

The input image is processed through various convolutional layers that are designed to extract features. These layers apply a set of filters to the input image to create feature maps. Each filter detects unique features at separate locations in the image in Fig 1.6.

1.5.2 Feature Maps of conv5 (Arbitrary Size)

This layer represents the output feature maps from the last convolutional layer (often referred to as "conv5" in Fig 1.6). The "arbitrary size" notation indicates that these feature maps can have any size, which is a property that the SPP layer can handle [19].

1.5.3 Spatial Pyramid Pooling Layer

The SPP layer takes the feature maps of varying sizes and applies the pyramid pooling operation. It pools the features in a way that generates fixed-size outputs regardless of the input feature map size. This is done by dividing the feature map into bins of different sizes (e.g., 16x256-d, 4x256-d, and 256-d) and then pooling the features in each bin [19] as shown in Fig 1.6.

After pooling, the SPP layer produces a fixed-length output, which is required for the next step in the network. Regardless of the original image or feature map size, the output is a flattened vector of a fixed length, which allows it to be fed into the fully connected layer.

1.5.4 Fully Connected Layers (fc6, fc7)

These layers further process the fixed-length vector from the SPP layer. They are standard layers in a neural network where every input is connected to every neuron, typically used for classification purposes. In Fig 1.6 the notation (fc6, fc7) corresponds to the naming convention of these layers in the network [19].

Figure 1.6 communicates the process by which an input image is transformed into a classifiable representation by the CNN, with the SPP layer ensuring that inputs of any size can be accommodated without loss of spatial information or need for input resizing. This feature makes the network more flexible and capable of handling a wide range of input dimensions.

1.6 Research Objectives

The main objective of this study is to build a model that effectively detects known and unknown malicious attempts against web applications. The proposed model is focused on the analysis of actual payload instead of header information in a packet, using deep learning technique. Considering the progress in image recognition technology, that have been leveraged to enhance the detection of malicious web traffic. By incorporating these techniques into this approach, anticipate notable enhancements in the detection capabilities, paralleling the advancements observed in image recognition technology. The payload data or URI queries have been converted into images format for model training.

1.7 Thesis Organization

The research thesis is organized in different chapters to provide the best possible understanding to the reader. Thus the following chapters have been constituted.

- Chapter 1 explores the background, problem statement, research motivation, web attacks, basic concepts, and objectives of the study.
- Chapter 2 studies the related research work.
- Chapter 3 covers the research methodology, dataset creation, and experimentation.
- Chapter 4 throws light on the examination, analysis, and evaluation.
- Chapter 5 summarises the research by covering the conclusion and future recommendations.

1.8 Thesis Contributions

This thesis contributes to the security of web applications by introducing an innovative deep learning-based framework for the detection of malicious web traffic. A comprehensive dataset has been created that contains a diverse array of real web attacks that belong to multiple categories. The main contributions of this research are as follows:

1.8.1 Development of Deep Learning Model

An advanced deep learning model has been designed, leveraging convolutional neural networks integrated with SPP. This model classifies web traffic into malicious and benign, showing remarkable accuracy in detecting various forms of web attacks. The utilization of SPP enables the model to maintain the spatial hierarchy of features in images created from web requests, thereby ensuring the retention of critical information during the classification process.

1.8.2 Curated Dataset of Web Attack Types

To bridge the gap in existing research, a richly annotated dataset has been created, which captures a wide spectrum of web attack vectors. This dataset is structured to reflect real-world

scenarios, encompassing clean traffic as well as malicious payloads, categorized into distinct classes of web attacks such as XSS, RCE, SQL injection, directory traversal, SSRF, and open-redirection.

1.8.3 Empirical Model Evaluation

Empirical evaluation methods have been used to assess the model's performance. The evaluation is done through confusion matrices, classification reports, and learning curves, which collectively provide a transparent view of the model's efficacy in differentiating between benign and malicious web traffic.

1.9 Summary

This chapter introduced the background of the study and some main research topics, which provides a comprehensive understanding of common web attacks and their associated risks. The motivation of the research has been discussed, to detect attacks on the fly and ensure the best possible security for web applications. The study's main aim is to develop a model that can detect known and new malicious web attacks against web applications using a proactive approach. The model utilizes deep learning techniques to analyze payload data, inspired by the progress in image recognition technology. The next chapter will give a detailed overview of the relevant studies that focused on the malicious traffic detection.

CHAPTER – 2

LITERATURE REVIEW

This chapter expands upon the concepts introduced earlier, focusing on current techniques and strategies in the domain of web application security. This literature review brings together a wide array of academic research, covering the development of online threats, the efficiency of present-day security solutions, and the latest patterns in cyber attacks, as highlighted in prominent industry reports such as the OWASP Top-10 [2] and the MITRE ATT&CK framework [20]. Various security approaches are discussed, ranging from conventional protocols to innovative deep learning technologies, and their roles in addressing the dynamic challenges of internet security vulnerabilities. The goal of this review is to provide an in-depth insight into the present landscape of web application security and to explore forward-thinking approaches.

2.0 Threat Landscape

In recent years, the landscape of web security has shown a rise in threats, as evidenced by the data from Radware's cloud WAF service [21]. Between 2021 & 2022, there was a remarkable 128% increase in the number of web application transactions blocked, a rate that significantly outpaces the 88% growth observed from 2020 to 2022. This trend was particularly notable during the first three quarters of 2021 when the number of blocked transactions steadily increased. Although there was a slight decrease in the fourth quarter, the count remained higher

than any quarter in 2020. The year 2022 marked a further acceleration in this upward trend, underscoring an exponential growth in web applications and online API attacks. It is important to note that these transactions were blocked either by custom rules set or through automated detection based on signature rules and behavioral algorithms. In 2022, half of these blocked transactions were identified based on known malicious behaviors, highlighting the increasing sophistication of web attacks and the need for advanced cybersecurity measures. Figure 2.1 illustrates the trend in web application attacks from 2020 to 2022 [21].

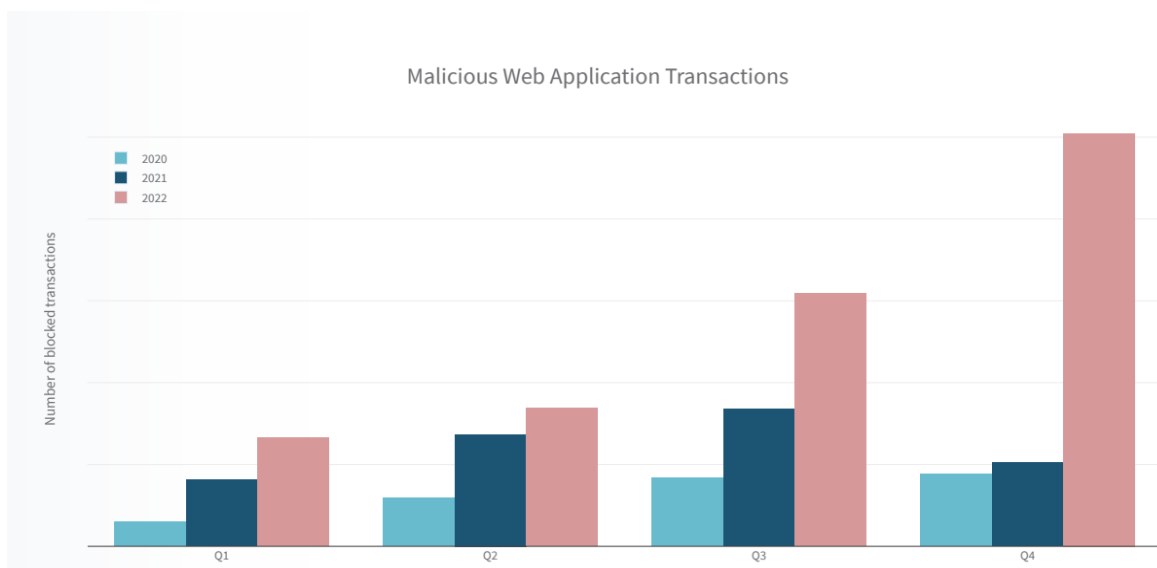


Figure 2.1 Trend in Web Application Attacks from 2020 to 2022 [21]

2.0.1 Industries Targeted by Cyber Attacks

In 2022, the industries that faced the highest number of cyber attacks included retail and wholesale trade, which comprised 25.3% of the incidents shown in Fig 2.2. This was closely followed by the high-tech industry at 19.5%, and telecommunications at 15.2% of these attacks. Collectively, these three sectors represented a massive portion, approximately 60%, of the web application attacks that were successfully thwarted [21].

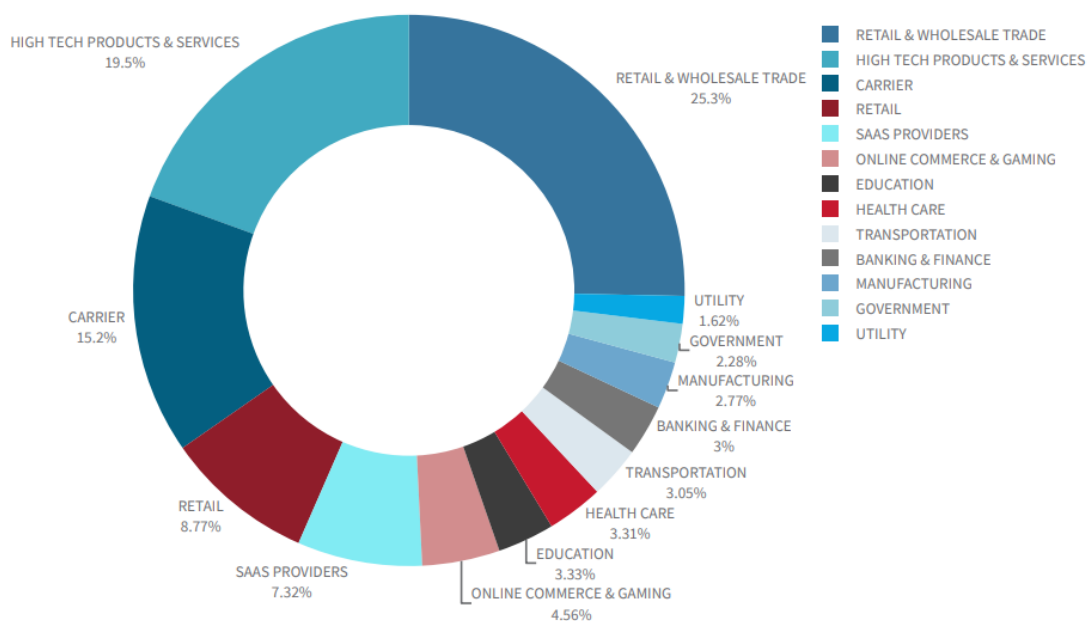


Figure 2.2 Top Attacked Industries [21]

2.0.2 Top Attacking Countries

A notable observation from 2022 is that a considerable proportion of web security incidents, 48.4%, were traced back to the United States [21]. Other countries including India, Italy, Russia, Netherlands, Canada, Germany, the United Kingdom, France, and Japan closely followed these nations in terms of number of attacks as shown in Fig 2.3.

However, the apparent source of a cyber attack does not always reveal the identity of the perpetrator. Cyber criminals frequently employ tactics such as VPNs, dark net routing, and the use of compromised systems in other countries to mask their activities. The choice of a country from which to launch an attack is often strategic and depends on factors like the location of the intended target or the desire to mislead investigations, which are known as false flag operations. This complexity adds another layer of challenge in accurately combating cyber threats.

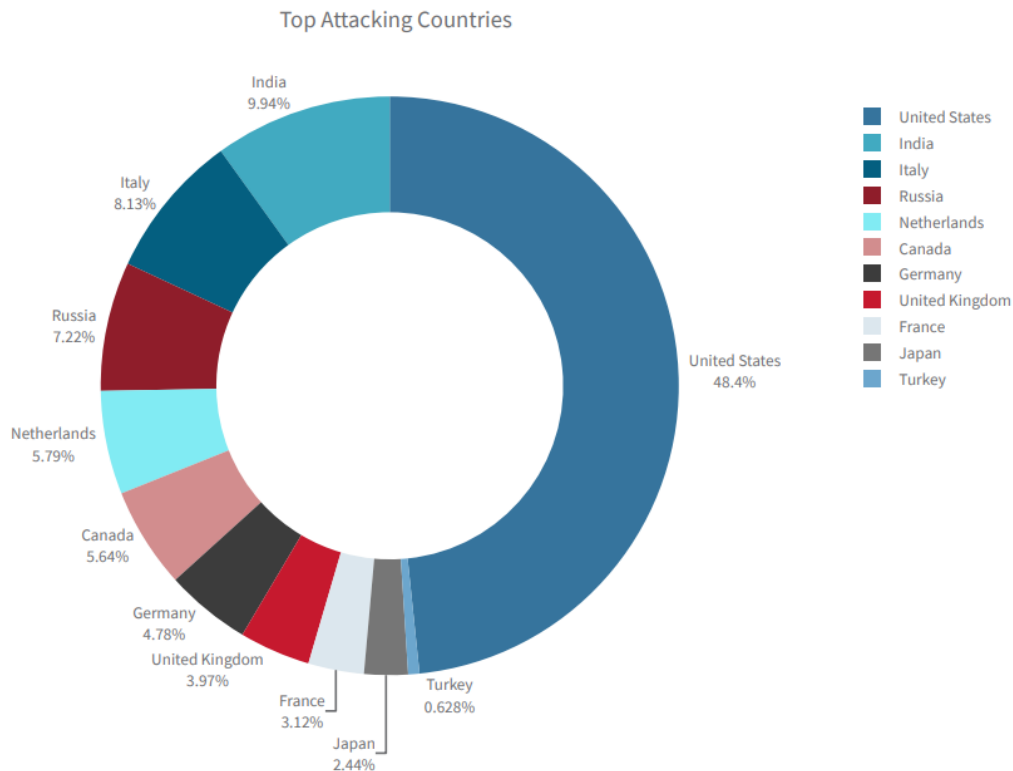


Figure 2.3 Top Attacking Countries [21]

2.0.3 Top Violation Types

Predictable resource location attacks stood out in all other violations in 2022 which constituted nearly half of all recorded attacks during the year as shown in Fig 2.4. Attackers engage in educated guessing of directory or file names, potentially gaining access to private resources. These resources could range from backup files and improperly secured configuration files to elements of a web application that are outdated, unpublished, or forgotten.

Followed by code injection 14.4% and SQL injection 10.9% attacks emerged as the most frequently employed techniques by cybercriminals targeting web applications and APIs.

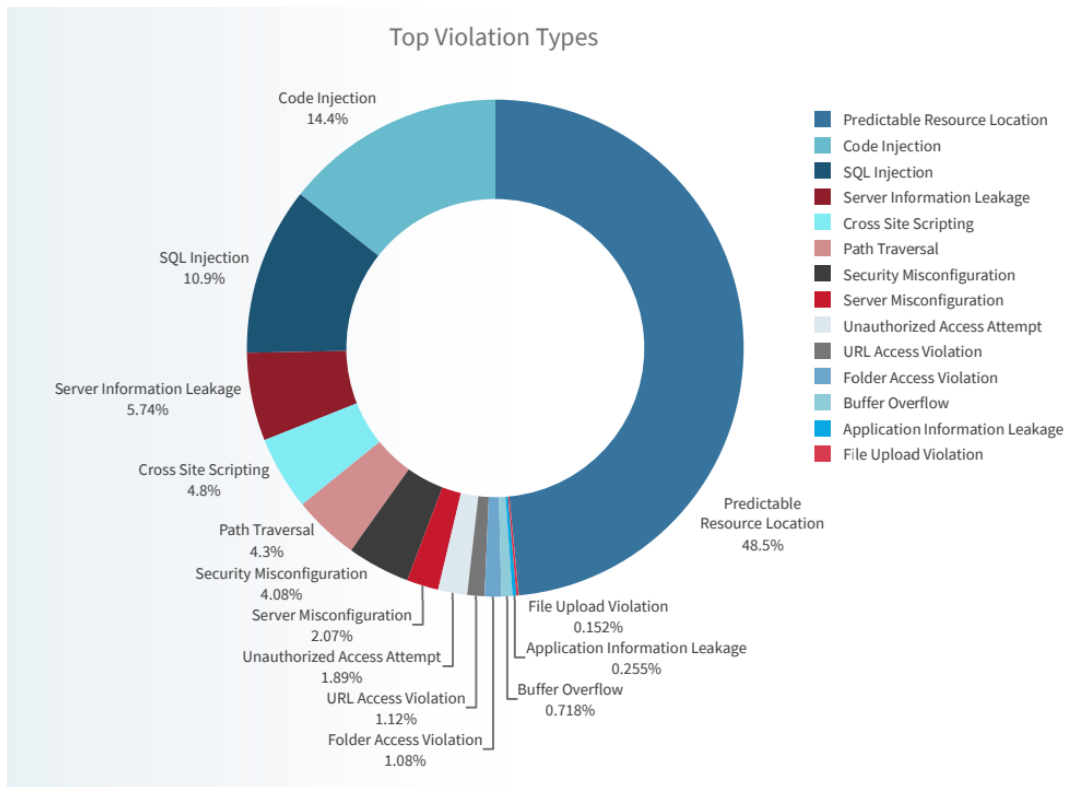


Figure 2.4 Top Violation Types [21]

2.1 Existing Approaches in Web Security

For web application security multiple approaches were used to efficiently mitigate a range of advanced attacks. This section of the literature review explores various techniques and best practices identified as effective in preventing common web application attacks.

2.1.1 Application Level Mitigations

- Input Validation and Sanitization** - a fundamental defense mechanism against attacks like XSS, SQL injection (SQLI), and command injection is careful input validation and sanitization. This involves scrutinizing user input to ensure it does not contain malicious scripts or commands. By treating all user input as untrusted, applications can prevent the execution of harmful scripts or database commands [22].
- Content Security Policy (CSP)** - implementing CSP is particularly effective against XSS attacks. It allows web applications to specify which sources are trusted, thereby preventing browsers from executing scripts from unauthorized sources [23].

- **Use of Parameterized Queries** - to combat SQLI, the use of parameterized queries is recommended. This technique ensures that the database interprets the input as data rather than executable code, thereby neutralizing the threat of malicious SQL code injection [24].
- **Least Privilege Access Controls** - implementing least privilege access controls is crucial in mitigating risks associated with SQLI, command injection, and remote code execution (RCE). Restricting the access rights of users and applications to the bare minimum necessary will reduce the potential damage from an attack [25].
- **Regular Software Updates and Patching** - keeping all software components up to date is critical in protecting against RCE and LFI attacks. Regular updates ensure that known vulnerabilities are patched, reducing the attack surface [26].
- **Network Segmentation and Firewalls** - in the context of SSRF and RCE and malware attacks, network segmentation and the use of firewalls can limit the scope of an attack. By segregating various parts of the network, attackers can be prevented from moving laterally [27].
- **File Access Restrictions** - to prevent directory traversal and local file disclosure (LFD) attacks, it is essential to enforce strict file access restrictions. This includes setting appropriate file permissions and ensuring that web applications do not expose sensitive file paths [28] & [29].
- **Whitelisting and Secure File Handling** - employing whitelisting for file inclusion and redirects can effectively mitigate LFI and open redirection attacks. This ensures that the application only allows access to or redirects to known, safe locations [30] & [31].
- **Disabling Unnecessary Features** - in the case of XML external entity (XEE) attack, disabling unnecessary features in XML processors, such as external entity processing, can close off this attack vector [32].

- **Secure Deserialization Practices** - to guard against insecure deserialization, it is recommended to avoid de-serializing data from untrusted sources. Where deserialization is necessary, implementing integrity checks like digital signatures can ensure the authenticity and integrity of the serialized data [33].

A security plan that includes these methods is required for a strong defense against various kinds of attacks. Regularly checking the security of the system, keeping an eye on it continuously, and following good security practices enhance these technical steps. Together, they form a key part of an active approach to keeping web applications secure.

2.1.2 Machine Learning Methods

A recent research [1] presents a groundbreaking machine learning-based framework for the detection and classification of such threats. The key aspect of the framework is the extraction and analysis of payloads from HTTP requests. The framework contains feature engineering and term weighting methods, with a focus on n-gram-based character level extraction. This approach enhances the model's ability to detect anomalies in web traffic. The study evaluates the efficacy of three classification algorithms, including support vector machine (SVM), random forest (RF), and stochastic gradient descent (SGD) against the dataset to determine their effectiveness in classifying various web attack types. The finding highlights the potential of SVM in cybersecurity applications, particularly for detecting and classifying web-based threats. The limitation of this work is the manual feature extraction for a diverse range of attacks, which is time-consuming and requires high pre-processing to train the model.

In [34] a new method, common attack pattern enumeration and classification (CAPEC) suggested by the authors that introduced a novel approach to web attack detection and classification. The innovative method of feature extraction was introduced based on ASCII values. It translates complex web requests into a numerically encoded format for machine

learning models. The author also developed the SR-BH 2020 multi-label dataset. Multiple algorithms, LightGBM and CatBoost were used against the new dataset. The whole web requests are converted to ASCII values which contain the URL and host part as well. These parts of the request don't add value to the detection and classification. This leads to a reduced accuracy of 88.44%.

M. Shah in [35] proposed a comprehensive framework for mitigating distributed denial of service (DDoS) attacks. This research focused on a dual approach, combining proactive and reactive strategies. The proactive approach is the adoption of a secure software development life cycle (SDLC) to ensure that security assurance activities, including penetration testing, code review, architecture analysis, etc. For reactive measures, a multi-layered defense strategy is employed including a DDoS dedicated solution, endpoint security firewall, perimeter firewall, load balancer, monitoring IPS/IDS, etc. This approach is not focused on zero-day attack that needs an adaptable solution.

2.1.3 Deep Learning Methods

A comprehensive study [36] addressed the critical challenge of network traffic classification, a task gaining popularity due to the exponential growth in network applications and the resultant surge in network traffic. This classification is essential for network operators to ensure quality of service (QoS) for various applications. The research used the deep learning method, specifically convolutional neural network (CNN) and residual network (ResNet), to classify network traffic, marking a significant advancement in the field. A novel approach to dataset preprocessing is introduced, converting packet payloads into image data for deep learning model training. This method is applied to a substantial dataset provided by the broadband communications research group. CNN is renowned for its data extraction capabilities, while ResNet introduces the innovative concept of shortcut connections, enhancing performance and optimization.

A novel approach to malware traffic classification, by using a representation learning approach using convolutional neural networks (CNN) has been proposed in [37]. This method makes a distinction from traditional traffic classification techniques, focusing on the direct use of raw traffic data as input for the classifier, thereby eliminating the need for manual features. The key aspect of the study is the selection of traffic granularity and packet layers for analysis. The paper discussed the use of different traffic split granularities, such as TCP connection, flow, session, service, and host, and opted for flow and session, which are commonly used in research. This representation learning technique, allows the classifier to learn features automatically from the raw data. The method involves using only the first n bytes of each flow or session, which may not be an effective approach if the later packets contain malicious content.

The research presented in [19] introduced an innovative methodology that combines deep convolutional layers with spatial pyramid pooling to address the challenge of handling images with varying sizes and aspect ratios. They replace conventional pooling layers with spatial pyramid pooling, which is a multi-level grid partitioning method, that allows to capture of features at different scales and aspect ratios. This adaptability enhances the network's precision and versatility in tasks such as object recognition and image analysis.

D-PACK, an unsupervised deep learning-based anomaly detection system for network traffic, was introduced in [38] to address security threats against the Internet of Things (IoT). With CNN and auto-encoders, D-PACK efficiently classified network flows into benign or malicious categories while using a threshold mechanism based on mean squared error (MSE Loss) distributions. This approach can be effective in many cases but it may not be suitable for handling highly dynamic or evolving network environments where the characteristics of benign traffic can change over time.

In [39] payload embeddings, a model was developed for intrusion detection, which was inspired from word2vec models used in natural language processing. Word2vec is made up of continuous bag of words (CBOW) and skip-gram models that help computers understand the meaning of words. Payload embeddings aims to understand network traffic data by creating special "embedding" for bytes and payloads in network packets. These embeddings are compact summaries that capture important information. The skip-gram model is used to create embeddings for individual bytes in the network packets and then feature vectors are created for the entire network packet payload. These vectors are used to classify and detect intrusions or unusual behavior in the network. Although this technique improved the accuracy of the existing solution, this approach is susceptible to attacks that primarily manipulate or exploit packet header information, such as scanning or probing attacks.

The authors in [40] propose an innovative solution through the implementation of spatial pyramid pooling in deep residual networks for the classification of malicious code. Codes are represented in images. This approach allows the processing of images of varying sizes without resizing to a fixed size to avoid information loss. The findings of this study indicate a notable improvement in classification accuracy and recall compared to conventional methods, marking a significant improvement in the field of network security and malware detection.

Kumar and Ponsam [41] propose an approach for the detection of Cross-Site Scripting (XSS) vulnerabilities. They proposed deep learning algorithms, Long Short Term Memory (LSTM), and Convolutional Neural Networks (CNN), along with boosting algorithms such as AdaBoost and Gradient Boosting. This comprehensive methodology enhanced the detection accuracy of XSS attacks. The scope of this work is limited to XSS only.

In [42] Vartouni introduced an innovative anomaly detection method using deep learning in stacked autoencoders (SAE), specifically designed for web attacks. Their approach combines

deep neural networks with isolation forests. By focusing on feature extraction and anomaly detection, this study contributes to the development of more accurate and efficient systems for identifying web-based threats. The paper discusses the generalization performance of the model but does not provide insights into how well the model can adapt to different types of web applications and traffic patterns.

Vyas et al. in [43] address the burgeoning field of IoT network security. They propose an improved Intrusion Detection System (IDS) utilizing deep learning for anomaly detection. By integrating a Convolutional auto-encoder for deep feature extraction, their model enhances the capability of IDS in IoT environment.

Dr. E. Amoroso presents Cyberlytic in [35], a tool designed to enhance web application security. It addresses the limitations of conventional web application firewalls (WAFs) by employing machine learning systems for dynamic threat identification. This tool is adopted to detect zero-day attacks and polymorphic variants, which traditional WAFs often miss due to their reliance on static signature and regular expression matching.

A new approach was proposed in [44] using deep learning technique that focuses on specific parameters within HTTP requests, including URL, user-agent, accept-language, connection, content length, and payload. HTTP anomaly detection is categorized into two principal domains, stream-based and payload-based. Stream-based anomaly detection leverages statistics derived from HTTP header and traffic data, primarily suited for identifying traffic-related attacks. Payload-based anomaly detection focuses on the content of HTTP packets. This study highlights the significance of URL and payload parameters due to their susceptibility to manipulation by both internal and external threats.

2.2 Summary

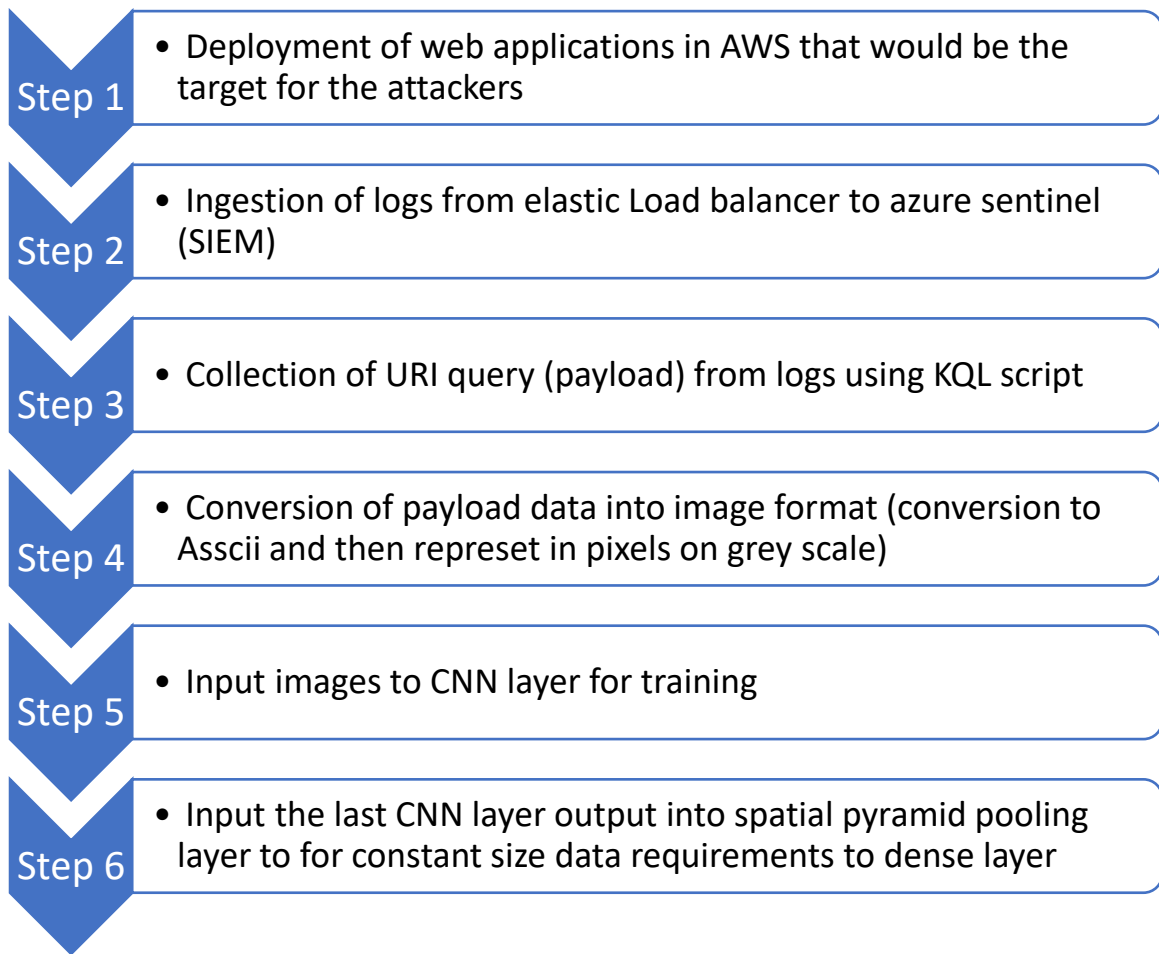
This chapter has examined various security approaches, from conventional protocols to innovative deep learning technologies. It aims to provide comprehensive insights into the current landscape of web application security and explores future progressive approaches. Key aspects include the escalation in web security threats, targeted industries, top attacking countries, violation types, and existing mitigation strategies across application-level defenses, machine learning, and deep learning level defenses. The following chapter will presents the research methodology and experimentation setup used in this thesis work.

RESEARCH METHODOLOGY

This chapter covers the research step-by-step process. A mix of deep learning and spatial pyramid pooling technique was proposed to mitigate web application attacks. Some web applications were deployed in a way that would attract these attacks, just like how a honeypot attracts bees. Then logs collected from these applications especially the parts that could provide indication of potential attacks. The collected data changed into a form that the deep learning program could understand. The goal is to make a CNN model mature enough to catch attacks against web applications.

3.0 Research Methodology Workflow

The following workflow provides an overview of the procedures, techniques, and processes employed in this thesis. The procedure begins with the deployment of web applications that serve as honeypots. Next, at load balancers web requests (logs) will be collected, which are subsequently ingested into Azure sentinel to extract crucial data. Following extraction, the payloads will be converted into image format in order to train the CNN model that will help categorize the payloads as clean or malicious.



3.1 Experimental Design and Procedures

The experimental design and procedures are discusses in this section, that gives the practical explanation of the steps followed for implementation of the proposed methodology.

3.1.1 Deployment of Web Applications in AWS Cloud

As a foundational component of the research methodology, web applications were deployed within the AWS cloud environment to serve as sophisticated interactive honeypots, mimicking real-world systems while being designed to attract and log cyber-attack attempts [45].

3.1.2 Planning and Architectural Design

The deployment process starts with planning to ensure the architecture will not only simulate a typical web environment but also enable detailed logging and analysis of cyber

threats. The system was designed with scalability and high availability in mind to manage and analyze a significant volume of data.

3.1.2.1 AWS Account and Security Configuration

An AWS account was created, followed by the careful configuration of IAM roles and policies. This measure was crucial to secure access to AWS resources and to define the permissions necessary for deployment and monitoring activities [46].

3.1.2.2 Service Selection

AWS elastic beanstalk was used for its ease of deployment and scaling web applications and services. This choice was selected to make an automated environment that could handle the provisioning, load balancing, and application health monitoring without extensive manual intervention [46].

3.1.2.3 Application Code Deployment

The application code was deployed through Elastic Beanstalk, with proper configurations required for simulating the real-world application behaviors [47].

3.1.2.4 Database and Storage

Amazon RDS [48] has been used due to its robust management capabilities of relational databases, and amazon S3 was employed to handle the storage of logs and other static assets with scalability and data durability as key factors [49].

3.1.2.5 Load Balancing and Auto-scaling

An elastic load balancer was configured to distribute incoming traffic across multiple instances of the applications, ensuring reliability and resilience. The auto-scaling feature was used to maintain application availability and automatically adjust capacity in response to incoming application traffic [50].

3.1.2.6 Compliance with AWS Best Practices

The deployment process adhered to the AWS well architected framework to align with cloud best practices and ensure the integrity and reliability of the research environment [51]. This is guide designed to assist users in constructing and implementing a highly optimized cloud environment. The objective of the framework is to reduce/manage the risks. Following are the key points in this framework:

- **Operational Excellence** - this point emphasizes on the importance of managing and monitoring system operations to deliver significant business value. It guides for the continuous enhancement of processes and procedures. Core elements include the management and automation of changes, efficient response to system events, and the establishment of standards for routine operational tasks.
- **Security** - this part is dedicated to security of information and infrastructures. It covers essential areas such as the confidentiality and integrity of data, management of user privileges, protection of system networks, and the implementation of controls for the timely detection of security incidents.
- **Reliability** - focusing on system dependability, this point cover enabling the system to prevent and swiftly recover from disruptions. It encompasses fundamental aspects like initial setup, requirements that span across various projects, planning for effective recovery, and managing changes without compromising system stability.
- **Performance Efficiency** - this part covers the use of IT and computing resources to achieve optimal performance. It involves choosing the most suitable types and sizes of resources based on specific workload requirements, continuously monitoring performance, and adapting to maintain high efficiency.

- **Cost Optimization** - key considerations include gaining a thorough understanding of expenditure, selecting the right resource types, financial analysis and scaling resources in a manner that aligns with requirements without leading to fiscal waste.

3.1.3 Data Ingestion

In the data ingestion phase, the primary focus is to capture comprehensive logs from the web applications deployed on AWS. These logs are important as they contain the details of web requests, including potential attack vectors that the ELB encounters.

Logs Collection at ELB - the ELB is configured to log every web request it processes. These logs offer insights into regular traffic as well as cyber-attacks. The ELB captures various attributes of each request, such as the requester's IP, requested paths and query parameters that are crucial for the subsequent analysis [52].

Logs Ingestion to Azure Sentinel - once collected, the logs are systematically transmitted to Sentinel, Microsoft cloud native SIEM. Sentinel provides a centralized platform for logs, enabling advanced analysis and threat detection. Logs can be ingested via data connector Amazon Web Service [53].

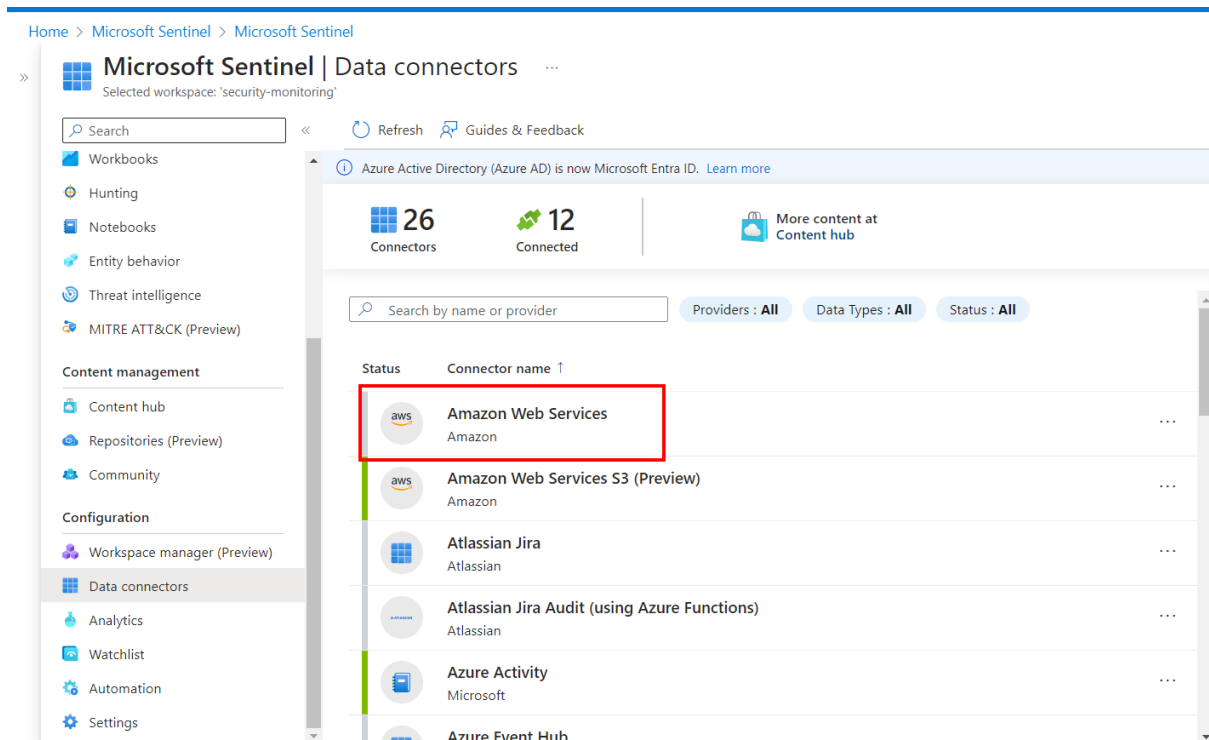


Figure 3.1 Logs Ingestion Connector

Filtering and Preprocessing of Logs - by default logs are not properly formatted in sentinel. To enhance efficiency and focus, filtering process were used, involves running multiple queries to scan through the vast data streams and extract only the payload part of the web requests [54] & [55]. Payloads, being the active content of web requests, often contain the most revealing signs of attempted or successful cyber-attacks. In Fig 3.2 sentinel's query were used to isolate these payloads based on string search using regex expression. This filtering significantly reducing the volume of information and enhancing the visibility of potential threats. This selective ingestion ensures that the datasets fed into the later stages are of high fidelity and relevance.

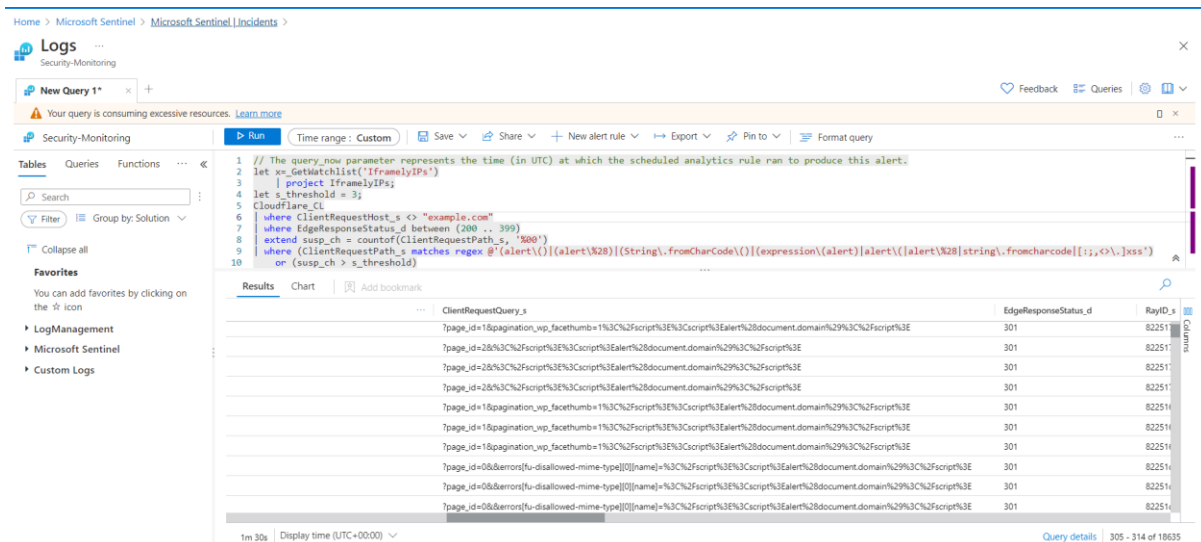


Figure 3.2 Kusto Query Language in Azure Sentinel for Malicious Request Collection

3.1.4 Data Labeling

Labeling Process - after filtering the logs for payloads, these were extracted into a CSV file for ease of use and accessibility. The labeling process was carried out manually, which involved a detailed review of the payload content to classify it to the specific category of malicious or benign.

Utilization of String Search in KQL - to enhance the precision of data collection, string searches were used within KQL. This helps to target specific patterns like SQLI (WHERE, FROM etc), XSS (Script, onload, onclick, onerror, alert etc) as shown in Fig 3.3. Then exported these filtered requests into CSV file and labeled them in chunks. By doing so, it also ensured that the collected data was not only relevant but also enriched with instances that are representative of real world cyber attack [56]. By thoroughly labeling the data, this established a groundwork for a model that is not only accurate but also capable of generalizing its detection capabilities to new, unseen cyber threats.

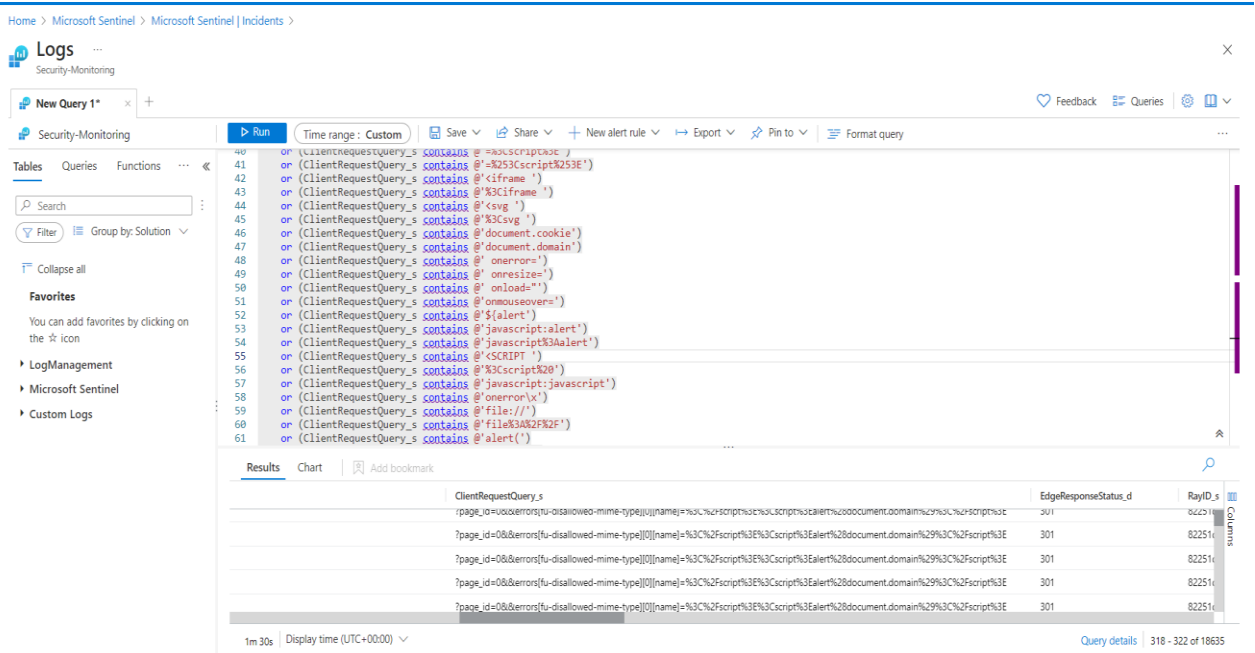


Figure 3.3 String Search in KQL

Dataset Insights - the result of this process was a curated dataset, encapsulated within a CSV file shown in Fig 3.4 ready for the next step. The file contains two columns, one column containing the payload and the second column is the corresponding label. This dataset was used for training the deep learning model to distinguish between malicious categories and benign web traffic with high accuracy.

6	RCE	313	313	2689	3315
7	SSRF	24	24	176	224
8	Total	11640	11640	93120	116400

Table 3.1 Dataset Composition

3.1.5 Data Conversion

Following the collection and labeling of payload data, the next crucial step is the data conversion process. This stage was designed to translate the textual payload data into a format that could be effectively processed by a CNN for pattern recognition and classification.

Conversion to ASCII - the process begins with the conversion of payload strings into their American Standard Code for Information Interchange (ASCII) numerical equivalents. Each character in the payload string was mapped to its corresponding ASCII value, thus transforming the textual data into a sequence of integers [57].

Pixel Representation - subsequently, the ASCII values are utilized to create a greyscale image representation of the data. Each ASCII value is translated into a pixel with a specific intensity on a greyscale. Intensity levels in a greyscale image can vary from 0 to 255, representing the spectrum from black to white, respectively as shown in Fig 3.5. Hence, the range of ASCII characters fit well within this spectrum, enabling a direct correlation between the character's ASCII value and the pixel's grayscale intensity [36]. All the process from text to image conversion was done with python automation as shown in Fig 3.6.

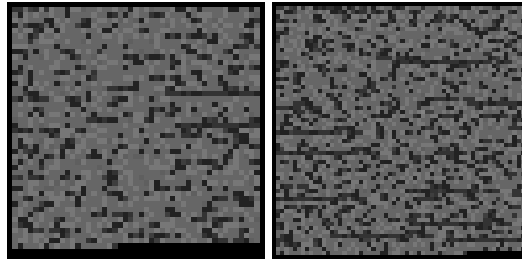


Figure 3.5 Greyscale Images

```

1  from PIL import Image
2  import numpy as np
3  import os
4  import csv
5
6  # Define the directory to save images
7  save_dir = "dataset"
8
9  # Read requests and labels from CSV file
10 with open('samples.csv', encoding='utf-8', errors='ignore') as csv_file:
11     csv_reader = csv.reader(csv_file, delimiter=',')
12     next(csv_reader) # skip header
13     for row_num, row in enumerate(csv_reader):
14         try:
15             request_str = row[0]
16             label = row[1]
17             # Convert the request string to ASCII values
18             ascii_values = [ord(c) for c in request_str]
19
20             # Calculate square and round up to the highest value
21             image_size = int(np.ceil(np.sqrt(len(ascii_values))))
22
23             # Add padding ASCII values (zeros) to make the total count a perfect square
24             padding_ascii_values = [0] * (image_size**2 - len(ascii_values))
25             ascii_values += padding_ascii_values
26
27             # Reshape the ASCII values into a 2D array
28             ascii_array = np.array(ascii_values, dtype=np.uint8).reshape((image_size, image_size))
29
30             # Calculate the new image size after adding padding
31             padded_size = image_size + 2
32
33             # Create the padded image array
34             padded_ascii_array = np.zeros((padded_size, padded_size), dtype=np.uint8)
35
36             # Copy the original ASCII values to the padded image array
37             padded_ascii_array[1:-1, 1:-1] = ascii_array
38
39             # Create the image using the ASCII values as pixel intensities
40             image = Image.fromarray(padded_ascii_array, mode='L')
41
42             # Save the image in the centralized directory with incremental filename
43             filename = f"{label}_{row_num}.bmp"
44             file_path = os.path.join(save_dir, filename)
45             try:
46                 image.save(file_path)
47             except SystemError:
48                 print(f"Skipping row {row_num}: image tile cannot extend outside image.")
49                 continue
50             print(f"Saved image {filename} in directory {save_dir}")
51         except UnicodeDecodeError:
52             print(f"Skipping row {row_num}: failed to decode request string.")
53             continue
54
55

```

Figure 2.6 Python Code for Text to Image Conversion

Greyscale Conversion - the choice to use greyscale images, as opposed to colored ones, was driven by the objective to reduce computational complexity without compromising the integrity of the information. Greyscale image contains 1 channel which reduce the computation of the model training [58]. Since the salient features in the payload data are not color-dependent, a greyscale format provides a sufficient level of detail. This simplification results in lower dimensionality, which in turn facilitates faster processing and analysis by CNN.

3.1.6 CNN Training

The training of the CNN model is the main step in proposed methodology, leveraging the transformed greyscale images as inputs to classify web traffic payloads effectively. The CNN model is structured to process variable input sizes and is constructed using Keras with TensorFlow as the backend.

3.1.6.1 Model Architecture

The CNN model is designed using the sequential model API in Keras [59], which allows for the linear stacking of layers as shown in Fig 3.7. Following is the model's architecture:

- The first layer is a Conv2D layer with 64 filters, and a kernel size (3x3). This layer uses the 'relu' activation function and is designed to take an input shape that allows for variable dimensions. Filter or kernel is a matrix with less number of rows and columns from the input image. The filter slides across the image with dot product operation to calculate the features [61].
- Following the initial convolutional layer, another Conv2D layer with 128 filters, utilizing the same kernel size and stride, again with 'relu' activation. This layer serves to further extract features from the input data. The rectified linear activation function, commonly known as “relu”, is a non-linear function. It outputs the input directly when it is positive; otherwise, it returns zero [62].

- The next layer in the sequence is also a Conv2D layer, this time with 256 filters, which continues the pattern of feature extraction at a higher level.
- A dropout layer has been used to avoid overfitting of the model. The Dropout layer functions like a filter, suppressing the impact of specific neurons on the next layer while allowing the unaffected neurons to maintain their usual functionality [63].
- To handle the variable input sizes and to avoid information loss, a spatial pyramid pooling (SPP) layer is integrated. The SPP layer pools the features at different regions and scales them, represented by [1, 2, 4], effectively allowing the model to maintain spatial hierarchies and adapt to various input dimensions.
- The final layer is a dense layer (Also called spatial fully connected layer) [64] with 7 units, corresponding to the number of classes used to predict. It uses a 'softmax' activation function [65] to output a probability distribution over the 7 classes.
- After evaluating the model for different epochs, the model has been trained for 25 epochs.

```

▶
model = Sequential()
model.add(Conv2D(64, (3, 3), activation='relu', input_shape=(None, None, 1)))
#model.add(MaxPooling2D((3, 3)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(Dropout(0.5))
model.add(SpatialPyramidPooling([1, 2, 4]))
model.add(Dense(7, activation='softmax'))

model.compile(optimizer=Adam(learning_rate=0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
#metrics=self.metrics)

model.summary()

model.fit(train_dataset,
          #steps_per_epoch=12,
          epochs = 25,
          validation_data = validation_dataset)

```

+ Code + Markdown

Figure 3.7 CNN Model Architecture

3.1.6.2 Optimization and Compilation

For optimization, Adam optimizer [66] has been used with a learning rate of 0.001 [67]. Adam is an optimization algorithm designed specifically for training deep neural networks. The loss function used is categorical cross-entropy which is appropriate for classification problems [68]. The model is compiled with the selected optimizer, and loss function, and includes 'accuracy' as the metric for performance evaluation.

3.1.6.3 Training Process

The training process involves feeding the prepared dataset of greyscale images into the CNN. The images, now in a standardized format, pass through the network's layers, where convolutional filters extract features and the SPP layer ensures that the feature maps are pooled to a fixed length. The training is conducted on a split of the dataset, with 10% testing, 10% validation, and 80% training.

The CNN's training phase is crucial for the model to learn the distinguishing features of malicious and benign payloads. This learning is reflected in the model's ability to generalize from the training data to accurately classify new, unseen data, which is the ultimate test of its efficacy.

3.2 Experimental Network Diagram

This experimental network diagram shown in Fig 3.8 illustrates the infrastructure utilized for dataset generation, encapsulating the entire flow of research methodology from beginning to end.

Initially, a user regular/attacker will navigate to the URL of the web application from their computer to access the service. The request is routed through the public internet to the web application hosted on AWS. Upon reaching AWS, the request first encounters the ELB, which acts as a distribution hub [69], directing the user's request based on the availability of the web

servers. These servers may exist across different availability zones for higher availability and fault tolerance.

For dataset generation, all user requests are collected at the load balancer by enabling logging functionality on the ELB. These logs are stored in an Amazon S3 bucket [70], which serves as a repository and is subsequently forwarded to Azure sentinel for enhanced monitoring and visibility. In Azure sentinel KQL is used for log analytics to extract specifically the URI query part from the requests, as this portion contains the actual request/query that a user sends to the application.

Once these URI queries exported into a CSV file a python script is used to convert these requests into images. This conversion process involves translating the text-based requests into ASCII format and subsequently into greyscale pixel values. The resulting collection of images forms the dataset that is used to train the CNN model, with the ultimate goal of identifying and categorizing web application attacks.

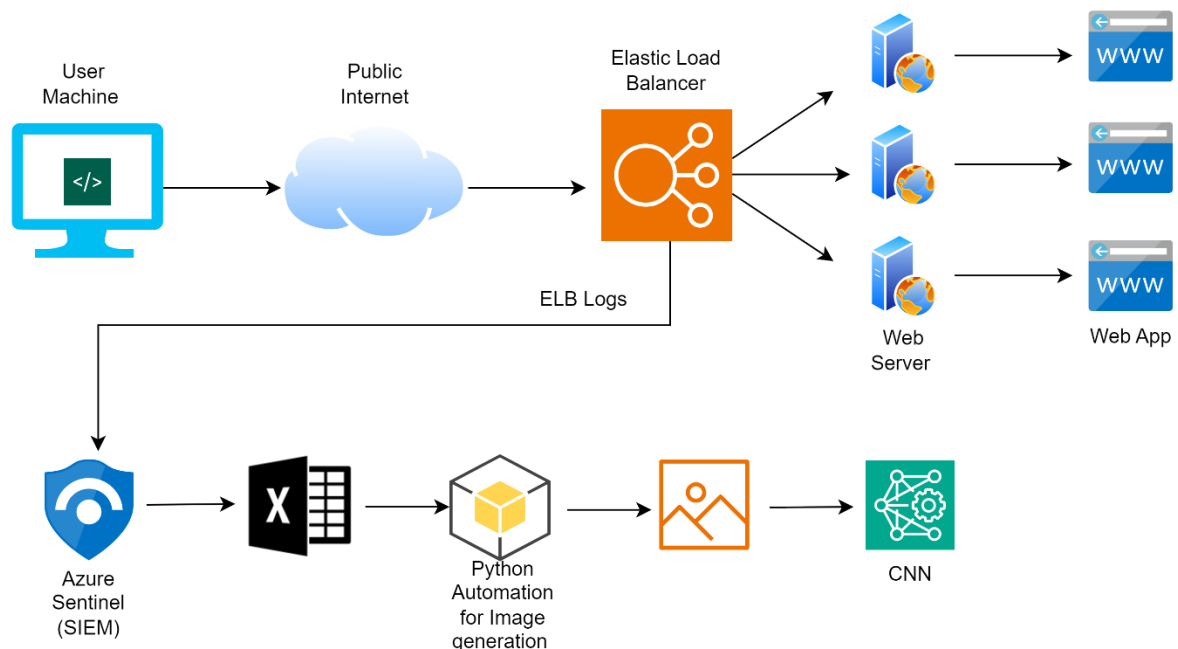


Figure 3.8 Experimental Network Diagram

3.3 Proposed Network Design

The proposed network diagram is the enhanced security infrastructure following the training of the CNN model as shown in Fig 3.9. Mirroring the experimental setup in its core design, this proposed system introduces a web application firewall into the network. This WAF serves as a gatekeeper, utilizing the intelligence derived from the trained CNN model.

WAF is positioned to evaluate incoming requests in real time. It applies the detection patterns learned by the CNN model to classify whether a request is benign or malicious. If a request is classified as malicious, the WAF will block that IP address immediately, thus preventing it from reaching the web servers.

The legitimate requests that pass through the WAF are then processed by the web servers, for the requested content.

The addition of the WAF in this proposed design is a significant progress towards an intelligent web application protection mechanism, which use the integration of an advanced AI model into a traditional network security framework.

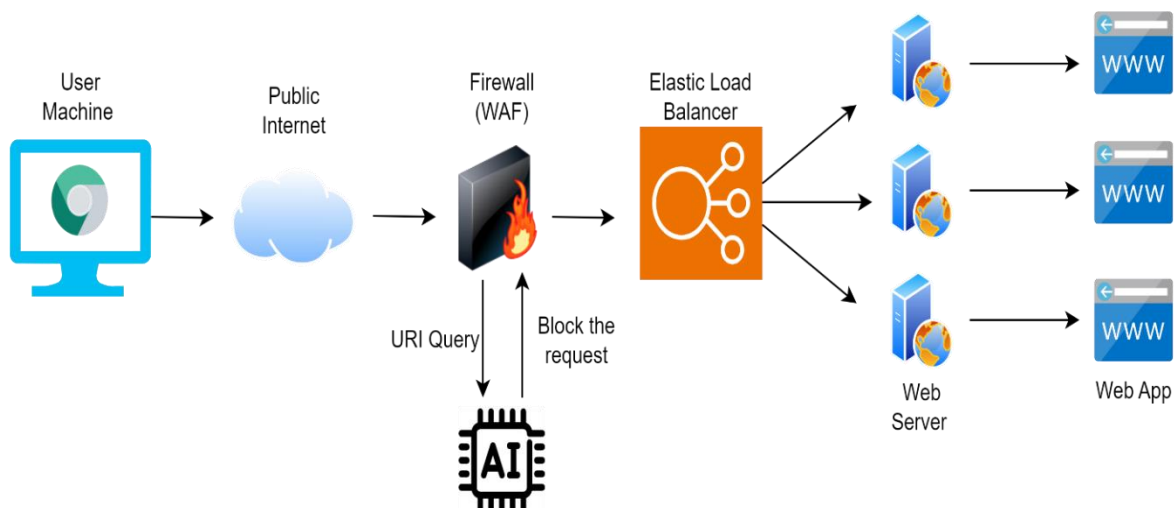


Figure 3.9 Proposed Network Diagram

3.4 Device Specifications

The specification of the computing device and tools that are used in this thesis are listed in

Table 3.2.

Property	Description
Architecture	X64
Processor	NVIDIA Tesla P100 GPUs
Core Count	3584 Cuda Cores
RAM	16 GB
Cloud	AWS
Python	Python3.0
AI Library	TensorFlow 2.13.0, Keras 3
SIEM	Azure Sentinel

Table 3.2 Specification of the Computing Device

3.5 Summary

This chapter covered a detailed description of the research methodology used to identify and prevent cyber threats in a cloud computing environment. It covers the development and deployment of web applications on AWS, the collection and storage of web traffic logs via ELB, and the utilization of Azure Sentinel for log analysis. Key to the methodology is the transformation of web traffic data into a format suitable for deep learning analysis, where a CNN model is trained to detect malicious activities. The trained CNN model is integrated into WAF to proactively block potential cyber-attacks, improving the security of web applications. The subsequent chapter will explain the examination and evaluation of the proposed model.

EXAMINATION AND ANALYSIS

After developing a deep learning model designed to detect malicious web traffic, it is crucial to understand how well it performs. This chapter explains the data gathered from the experiments, using metrics like accuracy, precision, and recall to evaluating the performance of the model.

4.0 Evaluation Metrics and Results Presentation

The effectiveness of the proposed deep learning model is measured using several performance metrics. Each metric offers unique insights into the model's predictive capabilities and shows how well the model can identify several types of web attacks.

4.0.1 Accuracy

This metric represents the ratio of correctly predicted instances to the total number of instances. Accuracy gives a quick indication of performance. A higher accuracy percentage signifies a greater reliability of the model in making correct predictions across the categories [71]. Overall, in Table 4.1 the model has an accuracy of 0.99, meaning it correctly predicted 99% of all classifications.

4.0.2 Precision

It is defined as the ratio of true positives (instances where the model correctly identifies malicious traffic) to the total number of instances labeled as positive by the model

(encompassing both true positives and false positives) [72]. The precision metric is particularly essential in this study as it gauges the model's proficiency in correctly pinpointing actual threats. A high precision score shows that the model flags traffic as malicious, there is a high likelihood of it being a true threat, thereby minimizing the risk of false alarms which can be crucial in real-world applications.

For example, in Table 4.1 the 'clean' traffic classification has a precision of 1.00, signifying that every instance predicted as 'clean' by the model was indeed clean.

4.0.3 Recall

Recall, alternatively known as sensitivity, measures the model's ability to identify all relevant instances. It is calculated as the ratio of true positives to the sum of true positives and false negatives (instances where the model fails to identify malicious traffic) [73]. In the domain of web security, recall assumes a critical role. It reflects the model's capacity to capture all potential threats, ensuring that malicious activities are not overlooked. A high recall score indicates that the model is effective in identifying the majority of actual threats, reducing the risk of dangerous oversights.

In Table 4.1, the recall of 1.00 for 'directory traversal' means that the model identified all instances of this attack in the dataset.

4.0.4 F1 Score

The F1 score as a critical metric in the model evaluation of the model, correlates the balance between precision and recall. It is the harmonic mean of these two metrics, offering a singular score that encapsulates both the model's accuracy in its positive predictions (precision) and its ability to comprehensively identify all relevant instances (recall) [74].

It serves as a more reliable metric of the model's performance, particularly in scenarios where an equilibrium between precision and recall is crucial. This balance is where overlooking

an attack (low recall) and misidentifying benign activities as threats (low precision) both carry significant consequences.

For example, in Table 4.1 F1-score of 1.00 for 'clean' traffic classification indicates perfect precision and recall.

4.0.5 Support

This number refers to the occurrence of the true responses in the dataset. For instance, there were 3,025 instances of 'clean' traffic in the test dataset.

4.0.6 Macro Average

The macro average calculates metrics for each label and finds their unweighted mean, by treating all classes equally. A macro average of 0.95 for recall suggests that the model performs uniformly across all labels, with some variance as indicated by less than 1 in Table 4.1.

4.0.7 Weighted Average

The weighted average considers each label's support. It is the average of the precision and recall of the model, weighted by the number of instances for each label. A weighted average of 0.99 for both precision and recall demonstrates the model's overall.

Category	Precision	Recall	F1-Score	Support/Sample Count
Clean	1.00	1.00	1.00	3025
Directory Traversal+LFI+LFD	0.99	1.00	1.00	3180
Injection	0.99	0.99	0.99	2785
Openredirection	1.00	0.99	0.99	248
RCE	0.94	0.93	0.93	313
SSRF	1.00	0.79	0.88	24
XSS	0.99	0.99	0.99	2065
Macro Average	-	-	0.99	11640
Weighted Average	0.99	0.95	0.97	11640
Accuracy	0.99	0.99	0.99	11640

Table 4.1 Performance Indicators

4.1 Confusion Matrix Analysis

The Confusion Matrix in Fig 4.1 is a measure of predictive analytics, offering a detailed breakdown of the model's predictions. Typically, a confusion matrix consists of four components, true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

True Positives (TP) - true positives occur when the model correctly predicts the positive class. In a web attack detection scenario, a TP would be an instance where the model correctly identifies a web request as malicious when it is actually malicious.

True Negatives (TN) - true negatives are cases where the model correctly predicts the negative class. In this work, a TN would be an instance where the model correctly identifies a web request as benign when it indeed is benign.

False Positives (FP) - false positives occur when the model incorrectly predicts the positive class. This would be a situation where the model identifies a web request as malicious when it is benign.

False Negatives (FN) - false negatives are cases where the model incorrectly predicts the negative class. This would be an instance where the model fails to identify a web request as malicious when it actually is malicious.

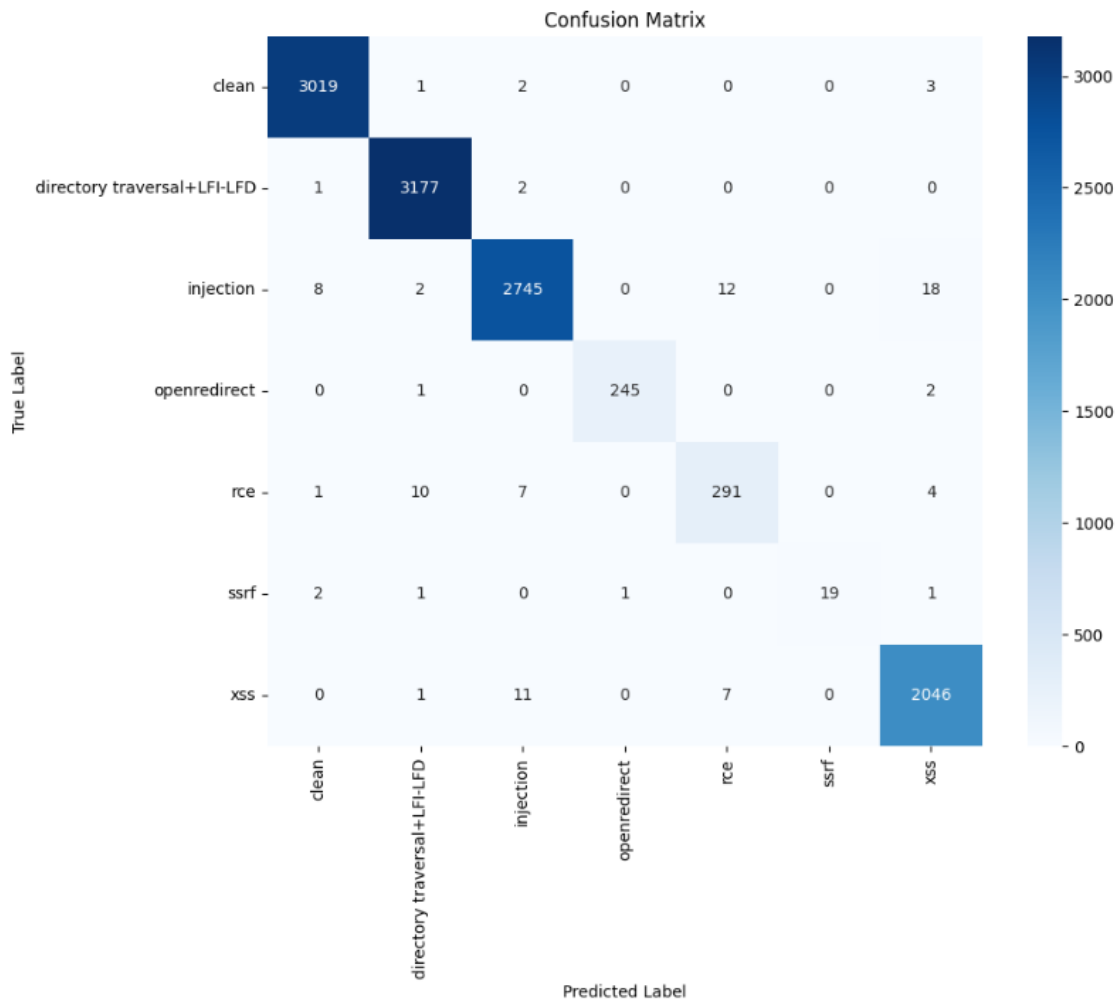


Figure 4.1 Confusion Matrix on Test Dataset

Within the matrix, the horizontal axis represents the predicted classifications made by the model, while the vertical axis corresponds to the true labels of the test data. The primary diagonal of the matrix, where the predicted label matches the true label, reveals the number of true positives for each attack category. For instance, the model has precisely identified 3,019 instances of 'clean' traffic, indicative of a robust ability to recognize benign interactions. Similarly, the classification of 'directory traversal' attacks is evidenced by 3,177 correct predictions in Fig 4.1.

Conversely, the off-diagonal elements illustrate instances where the model's predictions are incorrect, thus providing a measure of false positives and false negatives. These elements are

critical in identifying the types of errors the model is prone to, such as the 8 'injection' attacks incorrectly classified as 'clean', or the 7 instances where 'remote code execution' (RCE) attacks were misclassified as 'injection' attacks. Such misclassifications are indicative of areas where the model may require further refinement to enhance its predictive precision.

The intensity of color within the matrix visually represents the frequency of predictions, with darker shades corresponding to a higher number of occurrences for a given predictive outcome, and lighter shades showing a lower count.

4.1.1 Metrics Calculation Formulae

All these performance indicators will be calculated from the confusion matrix. Using Fig 4.2 as an example which shows the correct classification in green and incorrect classification in red. Performance metrics for the model is derived using the given formulas [75].

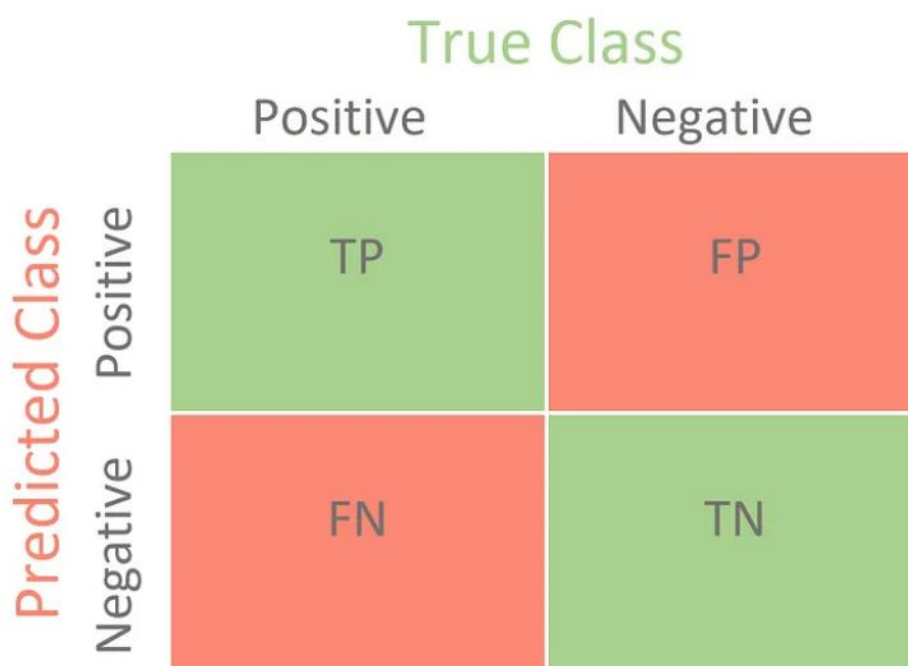


Figure 4.2 Metrics Calculation [61]

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

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

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

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

4.1.2 Accuracy and Loss Metrics across Epochs

Fig 4.3 presents a visual comparison of the proposed model's accuracy and loss during the training process. The 'model accuracy' graph on the left shows the classification accuracy of the model, with the training accuracy depicted in blue and the validation accuracy in orange. The 'model loss' graph on the right, illustrates the model's loss in predictions.

The training accuracy starts at an approximate value of 0.96 and increases within the initial epochs, stabilizing around 0.99. The validation accuracy closely trails the training accuracy, indicating that the model generalizes well to new data.

In the 'model loss' graph, a sharp decline in loss is evident for both training and validation during the early epochs. The training loss decreases rapidly and levels off, indicating that the model is effectively learning from the training data. The validation loss demonstrates a similar trend, with slight fluctuations, which is characteristic of the validation process as the model encounters previously unseen data. However, the model does not show an increasing trend in loss, which means that it is not under-fitting the data.

The graphs collectively demonstrate that the model's performance is consistent and reliable, with the validation metrics closely following the training metrics throughout the training process. This is indicative of a well-tuned model that is neither over fit nor under fit, but rather, is capable of accurately generalizing from the training data to predict unseen data.

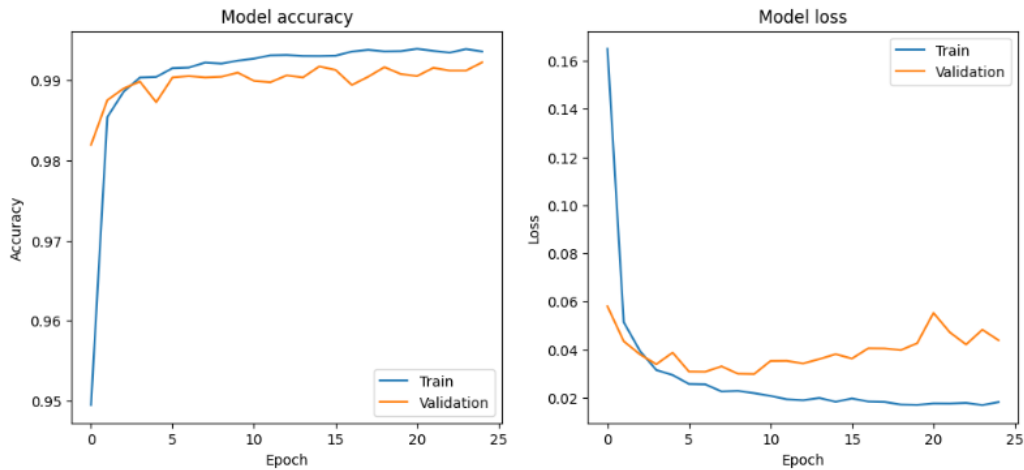


Figure 4.3 Model Accuracy and Loss Curves

4.2 Comparative Analysis

The proposed convolutional neural network with spatial pyramid pooling model, evaluated on the dataset of 116,400 instances, shows superior performance with accuracy, F1 score, recall, and precision all at an impressive 0.99. This outperforms the existing models in the literature such as LightGBM, CatBoost, SVM, Random Forest, SGD, and k-Nearest Neighbors [34]. The high precision and recall rates in Table 4.2 underscore the proposed model’s proficiency in classifying benign and malicious traffic, marking a significant advancement in the application of deep learning techniques in web application security.

Research work	Dataset Size	Algorithm	Accuracy	F1 Score	Recall	Precision
Tomás Sureda Riera [34]	907,814	LightGBM	0.88	0.88	0.88	0.90
Tomás Sureda Riera [34]	907,814	CatBoost	0.88	0.88	0.88	0.90
Shahin Ramezany [1]	111,721	SVM	0.98	0.85	0.80	0.92
Shahin Ramezany [1]	111,721	Random Forest’s	0.98	0.86	0.83	0.92
Shahin Ramezany[4]	111,721	SGD	0.97	0.84	0.80	0.92

Mehmet Engin Tozal [39]	95078	k-Nearest Neighbours	0.98	0.98	0.98	0.98
This research	116400	CNN-SPP	0.99	0.99	0.99	0.99

Table 4.2 Comparative Analysis

4.2.1 Categories-Based Accuracy

The dataset used in this thesis work contains seven categories. These categories are broadly classified in two classes clean and malicious. The malicious category further classified into specific web attack categories. The accuracy of the model for correctly classifying these categories show in Table 4.3 for each category.

Category	Accuracy %
Clean	99.80
Directory Traversal+LFI-LFD	99.91
XSS	98.56
Injection	98.79
Open-redirection	92.97
RCE	79.17
SSRF	99.08

Table 4.3 Categories-Based Accuracy

4.3 Summary

In this chapter, examination and analysis of the results obtained from the deep learning model, designed to detect malicious web traffic have been examined. A detailed evaluation of the model's performance, utilizing key metrics such as accuracy, precision, recall, F1 score, and a comprehensive analysis through the confusion matrix have been discussed in this chapter. The proposed model performed well among the existing machine learning and deep learning techniques with high accuracy, precision, and recall of 99% as compare 98% accuracy in achieved in [1] as shown in comparative analysis in Table 4.2. The model accuracy and loss curves shown in Fig 4.3 is indicative of a well-tuned model that is neither over fit nor under fit, but rather, is capable of accurately generalizing from the training data to predict unseen

data. The upcoming chapter will concludes the thesis with discussion on future work and recommendation.

CONCLUSION AND FUTURE HORIZONS

5.0 Conclusion

This chapter brings together the main points and the many outcomes of this research. Here a clear summarization of what has been learnt along with a discussion on possible future research. Also, the direction for future improvements have been identified.

5.0.1 Overview

This research presents a groundbreaking study focusing on the detection of malicious web traffic using a deep learning framework. By integrating convolutional neural networks with spatial pyramid pooling, this research proposed a novel approach to identifying and classifying web application attacks with high accuracy.

5.0.2 Key Findings

The developed model highlights remarkable performance metrics. Notably, the model achieved an accuracy of 99%, a precision of 99%, and a recall rate of 99%, as evidenced in the comparative analysis against established models. These metrics highlight the model's adeptness in identifying a range of web attack types, from XSS to SQL injection, with a low false positive rate. This efficiency shows the model potential as a robust tool for web security.

5.0.3 Significance

The integration of SPP within the CNN architecture enables the model to effectively handle variable payload sizes in the form of images, a common challenge in web traffic analysis. This capability is important to ensure the model's adaptability to different attack vectors.

5.0.4 Implications for Cyber Security

The success of this model in detecting malicious web traffic with high precision and recall opens new horizons in cybersecurity defense mechanisms against web applications. It provides a more accurate and effective approach compared to traditional methods, which often rely on signature-based detection.

5.1 Future Research Directions

While the current research has made significant strides in utilizing deep learning techniques, particularly in analyzing payload and URI queries for detecting web attacks, it is important to acknowledge its limitations in the context of certain types of cyber threats. Specifically, distributed denial of service (DDoS) and denial of service (DoS) attacks present unique challenges that are not fully addressed by a purely payload-based analysis approach.

Additionally, the exploration of the integration of this model with real-time monitoring systems for proactive threat detection and the inclusion of more diverse datasets could also be beneficial in improving the model's generalization capabilities.

To reduce the processing overhead, a formal method needs to be introduced that prevents the complete processing for same requests or duplicates.

REFERENCES

- [1] S. Ramezany, R. Setthawong and T. Tanprasert, "A Machine Learning-based Malicious Payload Detection and Classification Framework for New Web Attacks," 2022 19th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications, and Information Technology (ECTI-CON), Prachuap Khiri Khan, Thailand, 2022, pp. 1-4, doi: 10.1109/ECTI-CON54298.2022.9795455.
- [2] "OWASP Top Ten" [Online]. Available: <https://owasp.org/www-project-top-ten/> [Accessed April 2023].
- [3] "Network Flow Monitoring Explained: NetFlow vs sFlow vs IPFIX" [Online]. Available: <https://www.varonis.com/blog/flow-monitoring> [Accessed April 2023].
- [4] "Akamai Web Application and API Threat Report" [Online]. Available: <https://www.akamai.com/resources/research-paper/akamai-web-application-and-api-threat-report> [Accessed April 2023].
- [5] "Cross Site Scripting (XSS): What Is It & What's an Example?" [Online]. Available: <https://blog.hubspot.com/website/cross-site-scripting> [Accessed April 2023].
- [6] "Server-Side Request Forgery-SSRF" [Online]. Available: <https://www.briskinfosec.com/blogs/blogsdetail/Server-Side-Request-Forgery-SSRF> [Accessed May 2023].
- [7] "Structured Query Language Injection (SQLi)" [Online]. Available: <https://www.wallarm.com/what/structured-query-language-injection-sqli-part-1> [Accessed May 2023].
- [8] "Command Injection" [Online]. Available: <https://www.imperva.com/learn/application-security/command-injection> [Accessed May 2023].
- [9] "REMOTE CODE EXECUTION (RCE):PRINCIPLES AND FUNCTION " [Online]. Available: <https://www.crowdstrike.com/cybersecurity-101/remote-code-execution-rce/> [Accessed Aug 2023].
- [10] "Local file inclusion (LFI)" [Online]. Available: <https://www.invicti.com/learn/local-file-inclusion-lfi/> [Accessed Aug 2023].
- [11] "Local File Disclosure Vulnerability: A Case Study of Public-Sector Web Applications" [Online]. Available: <https://iopscience.iop.org/article/10.1088/1742-6596/933/1/012011> [Accessed Aug 2023].
- [12] "2017-XML External Entities (XXE)" [Online]. Available: https://owasp.org/www-project-top-ten/2017/A4_2017-XML_External_Entities_XXE [Accessed Aug 2023].
- [13] "What Is an Open Redirection Vulnerability and How to Prevent it?" [Online]. Available: <https://dzone.com/articles/what-is-an-open-redirection-vulnerability-and-how> [Accessed Aug 2023].

- [14] "Unvalidated Redirects and Forwards Cheat Sheet" [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html [Accessed Aug 2023].
- [15] "XML external entity (XXE) injection" [Online]. Available: <https://portswigger.net/web-security/xxe> [Accessed Aug 2023].
- [16] "INSECURE DESERIALIZATION" [Online]. Available: <https://www.contrastsecurity.com/glossary/insecure-deserialization> [Accessed Aug 2023].
- [17] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
- [18] "https://towardsdatascience.com/introducing-convolutional-neural-networks-in-deep-learning-400f9c3ad5e9" [Online]. Available: <https://towardsdatascience.com/introducing-convolutional-neural-networks-in-deep-learning-400f9c3ad5e9> [Accessed Sep 2023].
- [19] He, K., Zhang, X., Ren, S., Sun, J. (2014). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8691. Springer, Cham. https://doi.org/10.1007/978-3-319-10578-9_23
- [20] "ATT&CK Matrix for Enterprise" [Online]. Available: <https://attack.mitre.org/> [Accessed Sep 2023].
- [21] "2022 Global Threat Analysis Report" [Online]. Available: <https://www.radware.com/getattachment/b775fe9c-326f-4f97-b6c7-8545b5b68e42/Radware-2022-Global-Threat-Analysis-Report.pdf.aspx> [Accessed Sep 2023].
- [22] "Input Validation and Data Sanitization" [Online]. Available: <https://wiki.sei.cmu.edu/confluence/display/java/Input+Validation+and+Data+Sanitization> [Accessed Sep 2023].
- [23] I. Yusof and A. -S. K. Pathan, "Mitigating Cross-Site Scripting Attacks with a Content Security Policy," in Computer, vol. 49, no. 3, pp. 56-63, Mar. 2016, doi: 10.1109/MC.2016.76.
- [24] H.Fadlallah "Using parameterized queries to avoid SQL injection" [Online]. Available: <https://www.sqlshack.com/using-parameterized-queries-to-avoid-sql-injection/#:~:text=Parameterized%20queries%20is%20a%20technique,for%20the%20type%20and%20length.> [Accessed Sep 2023].
- [25] "principle of least privilege (POLP)" [Online]. Available: <https://www.techtarget.com/searchsecurity/definition/principle-of-least-privilege-POLP> [Accessed Sep 2023].
- [26] A. Hetler " 5 reasons software updates are important" [Online]. Available: <https://www.techtarget.com/whatis/feature/5-reasons-software-updates-are-important> [Accessed Sep 2023].

- [27] "Using parameterized queries to avoid SQL injection" [Online]. Available: <https://www.checkpoint.com/cyber-hub/network-security/what-is-network-segmentation/> [Accessed Oct 2023].
- [28] S. Mehnaz and E. Bertino, "A Fine-Grained Approach for Anomaly Detection in File System Accesses With Enhanced Temporal User Profiles," in *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, pp. 2535-2550, 1 Nov.-Dec. 2021, doi: 10.1109/TDSC.2019.2954507.
- [29] "File access control" [Online]. Available: <https://www.manageengine.com/device-control/file-access-control.html> [Accessed Oct 2023].
- [30] "Local File Inclusion (LFI): Understanding and Preventing LFI Attacks" [Online]. Available: <https://brightsec.com/blog/local-file-inclusion-lfi/> [Accessed Oct 2023].
- [31] "What is Local File Inclusion (LFI)?" [Online]. Available: <https://www.acunetix.com/blog/articles/local-file-inclusion-lfi/> [Accessed Oct 2023].
- [32] "How to Identify and Mitigate XXE Vulnerabilities?" [Online]. Available: <https://cybertrends-indusface.medium.com/how-to-identify-and-mitigate-xxe-vulnerabilities-a0ff56acaa07> [Accessed Oct 2023].
- [33] "Insecure deserialization" [Online]. Available: <https://learn.snyk.io/lesson/insecure-deserialization/> [Accessed Oct 2023].
- [34] Tomás Sureda Riera, Juan-Ramón Bermejo Higuera, Javier Bermejo Higuera, José-Javier Martínez Herraiz, and Juan-Antonio Sicilia Montalvo. 2022. A new multi-label dataset for Web attacks CAPEC classification using machine learning techniques. *Comput. Secur.* 120, C (Sep 2022). <https://doi.org/10.1016/j.cose.2022.102788>
- [35] M.Shah, "A SECURED AND ENHANCED MITIGATION FRAMEWORK FOR DDOS ATTACKS," 2019 <https://doi.org/10.26782/jmcms.2019.12.00075>
- [36] H. -K. Lim, J. -B. Kim, J. -S. Heo, K. Kim, Y. -G. Hong and Y. -H. Han, "Packet-based Network Traffic Classification Using Deep Learning," 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIC), Okinawa, Japan, 2019, pp. 046-051, doi: 10.1109/ICAIC.2019.8669045.
- [37] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye and Yiqiang Sheng, "Malware traffic classification using convolutional neural network for representation learning," 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam, 2017, pp. 712-717, doi: 10.1109/ICOIN.2017.7899588
- [38] R. -H. Hwang, M. -C. Peng, C. -W. Huang, P. -C. Lin and V. -L. Nguyen, "An Unsupervised Deep Learning Model for Early Network Traffic Anomaly Detection," in *IEEE Access*, vol. 8, pp. 30387-30399, 2020, doi: 10.1109/ACCESS.2020.2973023.
- [39] M. Hassan, M. E. Haque, M. E. Tozal, V. Raghavan and R. Agrawal, "Intrusion Detection Using Payload Embeddings," in *IEEE Access*, vol. 10, pp. 4015-4030, 2022, doi: 10.1109/ACCESS.2021.3139835.

- [40] J. Liu, Y. Qu, J. Li, Y. Wang, J. Zhang and H. Yin, "Malicious Code Family Classification Method Based on Spatial Pyramid Pooling and Deep Residual Network," 2021 IEEE 7th International Conference on Cloud Computing and Intelligent Systems (CCIS), Xi'an, China, 2021, pp. 260-264, doi: 10.1109/CCIS53392.2021.9754597.
- [41] J. Harish Kumar and J. J Godwin Ponsam, "Cross Site Scripting (XSS) vulnerability detection using Machine Learning and Statistical Analysis," 2023 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2023, pp. 1-9, doi: 10.1109/ICCCI56745.2023.10128470.
- [42] A. M. Vartouni, S. S. Kashi and M. Teshnehlab, "An anomaly detection method to detect web attacks using Stacked Auto-Encoder," 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), Kerman, Iran, 2018, pp. 131-134, doi: 10.1109/CFIS.2018.8336654.
- [43] M. Vyas, R. Vijayaganth, J. Chandhok, A. Srivastava, S. Arumugam and M. Tiwari, "Revolutionizing IoT Network Security with Deep Learning-Anomaly Detection Model," 2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2023, pp. 1533-1538, doi: 10.1109/ICESC57686.2023.10193493.
- [44] Tekerek, A. (2021). A novel architecture for web-based attack detection using convolutional neural network. *Computers & Security*, 100, 102096.
- [45] "What is Honeypot?" [Online]. Available: <https://www.geeksforgeeks.org/what-is-honeypot/> [Accessed Nov 2023].
- [46] "Identity and Access Management for AWS Account Management" [Online]. Available: <https://docs.aws.amazon.com/accounts/latest/reference/security-iam.html> [Accessed Oct 2023].
- [47] "Deploy and scale web applications" [Online]. Available: https://aws.amazon.com/elasticbeanstalk/?gclid=CjwKCAiA1fqrBhA1EiwAMU5m_2tdHnQhM577JDh9bfnTn1kBKg_5DZcr81dRP5Uc9VY6P_cD5Z8GjxoCsogQAvD_BwE&trk=29514334-a45b-4894-96c1-cd1eff3a5e50&sc_channel=ps&ef_id=CjwKCAiA1fqrBhA1EiwAMU5m_2tdHnQhM577JDh9bfnTn1kBKg_5DZcr81dRP5Uc9VY6P_cD5Z8GjxoCsogQAvD_BwE:G:s&s_kwcid=AL!4422!3!651510255291!e!!g!!elastic%20beanstalk!19836376744!146491721185 [Accessed Nov 2023].
- [48] "Create an Amazon RDS DB instance" [Online]. Available: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Tutorials.WebServerDB.CreateDBInstance.html [Accessed Nov 2023].
- [49] "What is Amazon Relational Database Service (Amazon RDS)?" [Online]. Available: <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html> [Accessed Oct 2023].
- [50] "EC2 Auto Scaling & Elastic Load Balancer | AWS in Action" [Online]. Available: <https://www.youtube.com/watch?v=cf9jQc4xzpo> [Accessed Oct 2023].

- [51] “Essential AWS Security Best Practices - YouTube” [Online]. Available: <https://www.youtube.com/watch?v=mPU43QvPAHE> [Accessed Oct 2023].
- [52] “Access logs for your Application Load Balancer” [Online]. Available: <https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-access-logs.html> [Accessed Nov 2023].
- [53] “Collect data in custom log formats to Microsoft Sentinel with the Log Analytics agent” [Online]. Available: <https://learn.microsoft.com/en-us/azure/sentinel/connect-custom-logs?tabs=DCG> [Accessed Oct 2023].
- [54] D. Vannoy “Querying Azure Log Analytics (with KQL)” [Online]. Available: https://www.youtube.com/watch?v=92oJ20XeQso&ab_channel=DustinVannoy [Accessed Nov 2023].
- [55] “Filter & Split Firewall/CEF logs into multiple Sentinel tables (analytics/basic tier) to save in ingestion costs” [Online]. Available: <https://www.linkedin.com/pulse/filter-split-firewallcef-logs-multiple-sentinel-tables-marko-lauren> [Accessed Nov 2023].
- [56] “Kusto Query Language (KQL) overview” [Online]. Available: <https://learn.microsoft.com/en-us/azure/data-explorer/kusto/query/inoperator> [Accessed Oct 2023].
- [57] “Convert String to ASCII” [Online]. Available: <https://mkyong.com/java/how-to-convert-character-to-ascii-in-java/#:~:text=Convert%20String%20to%20ASCII,to%20get%20the%20ASCII%20value.> [Accessed Nov 2023].
- [58] “GRAY SCALE IN CNN” [Online]. Available: <https://www.linkedin.com/pulse/gray-scale-cnn-kamalakkannan-r-a9bgc> [Accessed Nov 2023].
- [59] Gulli, A., Kapoor, A., Pal, S. (2019). Deep Learning with TensorFlow 2 and Keras: Regression, ConvNets, GANs, RNNs, NLP, and More with TensorFlow 2 and the Keras API, 2nd Edition. United Kingdom: Packt Publishing.
- [60] “Keras Model Sequential API VS Functional API” [Online]. Available: <https://medium.com/analytics-vidhya/keras-model-sequential-api-vs-functional-api-fc1439a6fb10#:~:text=Sequential%20API%20allows%20you%20to,have%20multiple%20inputs%20or%20outputs.&text=And%20now%20lets%20plot%20your%20model%20using%20keras%20utils.> [Accessed Nov 2023].
- [61] “Beginners Guide to Convolutional Neural Networks” [Online]. Available: <https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d#:~:text=Convolution%20of%20RGB%20image%20using,function%20in%20a%20convolution%20layer.> [Accessed Nov 2023].
- [62] “What is ReLU ?” [Online]. Available: <https://iq.opengenus.org/relu-activation/> [Accessed Nov 2023].
- [63] “How ReLU and Dropout Layers Work in CNNs” [Online]. Available: <https://www.baeldung.com/cs/ml-relu-dropout-layers> [Accessed Nov 2023].

- [64] “Keras Dense Layer: How to Use It Correctly” [Online]. Available: <https://wandb.ai/ayush-thakur/keras-dense/reports/Keras-Dense-Layer-How-to-Use-It-Correctly--Vmlldzo0MjAzNDY1> [Accessed Nov 2023].
- [65] “Softmax Activation Function — How It Actually Works” [Online]. Available: <https://towardsdatascience.com/softmax-activation-function-how-it-actually-works-d292d335bd78> [Accessed Nov 2023].
- [66] “Gentle Introduction to the Adam Optimization Algorithm for Deep Learning” [Online]. Available: <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/> [Accessed Oct 2023].
- [67] “What is the learning rate in Machine Learning?” [Online]. Available: <https://deepchecks.com/glossary/learning-rate-in-machine-learning/> [Accessed Nov 2023].
- [68] “Cross-Entropy Loss Function” File Disclosure Vulnerability: A Case Study of Public-Sector Web Applications” [Online]. Available: <https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e> [Accessed Nov 2023].
- [69] “Getting started with Elastic Load Balancing” [Online]. Available: <https://aws.amazon.com/elasticloadbalancing/getting-started/> [Accessed Nov 2023].
- [70] “Enabling Amazon S3 server access logging” [Online]. Available: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/enable-server-access-logging.html> [Accessed Nov 2023].
- [71] “Accuracy (error rate)” [Online]. Available: <https://deepai.org/machine-learning-glossary-and-terms/accuracy-error-rate> [Accessed Nove 2023].
- [72] “Precision and Recall | Essential Metrics for Machine Learning (2023 Update)” [Online]. Available: <https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/> [Accessed Nov 2023].
- [73] “What is a Confusion Matrix in Machine Learning?” [Online]. Available: <https://www.simplilearn.com/tutorials/machine-learning-tutorial/confusion-matrix-machine-learning> [Accessed Nov 2023].
- [74] “Confusion Matrix for Your Multi-Class Machine Learning Model” [Online]. Available: <https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826> [Accessed Nov 2023].
- [75] “Accuracy vs. precision vs. recall in machine learning: what's the difference?” [Online]. Available: <https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall#:~:text=Accuracy%20shows%20how%20often%20a,objects%20of%20the%20target%20class.> [Accessed Nov 2023].

