

# **Malware Detection using Graph Neural Network and comparison with Traditional Neural Network**

by

Nihal Nisar Parkar

Submitted to

**The University of Roehampton**

In partial fulfilment of the requirements

for the degree of

**MASTER OF SCIENCE IN COMPUTING**

### **Abstract:**

This paper explores various deep learning models, such as GraphSAGE, GCN, Hyper-parameter GCN, Graph Isomorphism Network, Graph Attention Network, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and GCN, for malware identification. Graph Isomorphism Network is the most accurate of them all, with 97.58% accuracy, indicating that it has the capacity to remove malware. A comparative examination of these models, revealing their differing degrees of correctness, is provided by the research, which advances cybersecurity. The development of more reliable malware detection systems will benefit greatly from these insights. The paper highlights Graph Isomorphism Networks as a prominent competitor in this field and emphasizes the promise of specialized graph-based neural networks in particular. This work improves understanding of deep learning in cybersecurity and highlights how important it is to choose appropriate models for particular tasks, which will aid in the creation of reliable malware detection technologies.

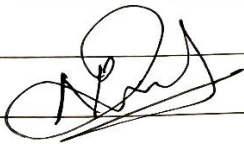
## Declaration

I hereby certify that this report constitutes my own work, that where the language of others is used, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions, or writings of others.

I declare that this report describes the original work that has not been previously presented for the award of any other degree of any other institution.

**Nihal Nisar Parkar**

**Date:** 20/05/2024

A handwritten signature in black ink, appearing to read 'Nihal Nisar Parkar', is written over four horizontal lines. The signature is fluid and cursive, with the first name 'Nihal' being the most prominent part.

Signed (apply signature below)

# Acknowledgments

I would like to express my profound gratitude to my esteemed advisor, Professor Muneer Ahmad, for steadfast support and advice during the dissertation's research and writing phase. Professor Muneer Ahmad always kept the lines of communication open and was a great help when I ran into problems with my research and writing. Furthermore, his guidance has been really helpful in fostering my intellectual development and making sure that this dissertation is successfully finished.

## Table of Contents

Chapter1: Introduction .....	7
1.1. Introduction .....	7
1.2. Motivation.....	8
1.3. Research Question.....	11
1.4. Research Objectives .....	11
1.5. Research Outline.....	11
Chapter 2: Literature Review .....	13
2.1 Concept of Malware Detection Using Deep Learning .....	13
2.2 Literature Review .....	14
2.3 Table of Summary .....	23
Chapter 3: Methodology .....	27
3.1 Dataset .....	29
3.2 Data Analysis using Tableau .....	30
3.3 Data Preparation .....	34
3.4 Data Visualization:.....	36
3.5 Data Modelling: .....	42
3.6 Model Used:.....	43
3.6.1. Convolutional Neural Network (CNN) .....	43
3.6.2. Recurrent Neural Network (RNN) .....	43
3.6.3. Graph Sample and Aggregated (GraphSAGE) .....	43
3.6.4. Graph Convolutional Network (GCN).....	44
3.6.5. Hyper-parameter tuning for GCN.....	44
3.6.6. Graph Isomorphism Network (GIN) .....	44
3.6.7. Graph Attention Network (GAT).....	44
Chapter 4: Result and Discussion.....	45
4.1. Precision, Recall, and F1 score .....	45
Chapter 5: Conclusion and Future scope.....	48
Chapter 6: References .....	50

## List of Figures

Figure 3. 1: CRISP-DM .....	27
Figure 3. 2: Project Methodology .....	28
Figure 3. 3: Dashboard 1.....	30
Figure 3. 4: Tableau Dashboard 2.....	31
Figure 3. 5: Tableau Dashboard 3.....	31
Figure 3. 6: Tableau Dashboard 4.....	32
Figure 3. 7: Tableau Dashboard 5.....	33
Figure 3. 8: Tableau Dashboard 6.....	33
Figure 3. 9: Tableau Dashboard 7 .....	34
Figure 3. 10: Import all libraries .....	35
Figure 3. 11: Read the dataset .....	35
Figure 3. 12: Checking the null values.....	36
Figure 3. 13: Statistical Description of Dataset.....	36
Figure 3. 14: Drop the unwanted columns .....	36
Figure 3. 15: Count plot for Malware.....	37
Figure 3. 16: Pair plot for first 5 columns .....	38
Figure 3. 17: Pair plot for next 6 columns .....	39
Figure 3. 18: Violin Plot for Malware Type and Major Linker Version .....	40
Figure 3. 19: Correlation Heat map 1.....	40
Figure 3. 20: Correlation Heat map 2.....	41
Figure 3. 21: Correlation Heat map 3.....	41
Figure 3. 22: Separating the Dependent and independent variable.....	42
Figure 3. 23: Perform train and test split.....	42
Figure 3. 24: Normalize the dataset.....	43
Figure 4. 1: Accuracy of the models .....	45

## List of Tables

Table 2. 1: Table of Summary .....	26
Table 4.1: Precision, Recall, and F1 score.....	47

## Chapter1: Introduction

### 1.1. Introduction:

Cybersecurity is being revolutionized by deep learning models, especially in malware detection. This work carefully investigates and assesses a number of sophisticated models designed for deep learning-based malware detection. Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), GraphSAGE, Hyper-parameter GCN, Graph Isomorphism Network, and Graph Attention Network are some of the networks in the ensemble. This study emphasizes the importance of these models in fending off changing threats and advances our understanding of the dynamic cybersecurity landscape.

The core of this work is an in-depth analysis of several models that have been painstakingly created to identify complex patterns and anomalies suggestive of hostile activity in digital systems. Examining accuracy measures reveals that the Graph Isomorphism Network is particularly effective, but the other models also add to our understanding by providing more detailed insights into their working mechanisms, thereby clarifying both their advantages and disadvantages.

Enhancing cybersecurity measures is the main objective of this research, which goes beyond technical details. Through a comparative investigation of multiple deep learning models for malware detection, it aims to create a solid framework for improving current detection methods. This paper provides a thorough analysis of several models and a detailed grasp of their capabilities, allowing for strategic decision-making according to the particular requirements of various cybersecurity projects.

These results highlight the promise of specialized graph-based neural networks, of which the Graph Isomorphism Network is a prominent example. They also clarify more general implications for cybersecurity deep learning. The study highlights the necessity for a customized strategy to successfully tackle the always changing malware landscape and highlights how crucial it is to use the appropriate model for particular activities. In summary, our work adds to the body of information already available on deep learning applications in cybersecurity and establishes the foundation for the development of more reliable and accurate malware detection solutions.

## **1.2. Motivation:**

Taking advantage of deep learning is imperative as the volume of digital data increases quickly and the threat of malware assaults becomes more real. The potential for combining deep learning and cybersecurity, as noted by Awan et al. (2021), is what spurs this study and motivates efforts to not only detect but also prevent malware threats. As malware strains get more complex, traditional tactics become less effective, which leads to a change in strategy. Modern methods must be flexible enough to adapt to changing threats in the ever-changing landscape of cybersecurity.

The inspiration also comes from realizing how unique deep learning models are at extracting relevant features from unprocessed data on their own. This ability comes in quite handy when trying to identify complex patterns in malware. Both Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are powerful tools that can reveal subtleties, which makes it easier to identify new threats like zero-day malware vulnerabilities.

The commitment to overcoming the challenges associated with integrating deep learning into malware detection is at the heart of our effort. The stability and reliability of these models remain critical issues, particularly in the face of hostile attacks that take advantage of flaws. Additionally, the reason includes the pursuit of a subtle balance between rigor and transparency in deep learning models and the recognition of the need to understand their decision-making processes.

There is a constant need to investigate new tactics to protect people, companies, and countries in the dynamic field of cybersecurity, which is typified by continuous malware attacks. A complex dynamic develops between cybersecurity experts and hackers as malicious software attacks spread around the globe. Deep learning and cybersecurity come together to offer a viable way to combat malware invasions, both proactively and reactively. In the face of this growing threat, this combination is encouraging.

The digital age has resulted in an unparalleled increase in data volume, challenging traditional malware detection techniques to keep up with the complexity of contemporary threats. The need for improved tactics is highlighted by the shortcomings of static, signature-based approaches in managing the wide variety and complexity of contemporary malware. In order to enhance intelligent malware detection systems, this research explores the intersection of deep learning and cybersecurity, looking into potential areas of mutually beneficial cooperation.



Academic and industrial experts are embracing a paradigm shift by combining neural networks and state-of-the-art architectures in an attempt to prevent harmful software. With the threat landscape changing, this strategic move acknowledges the diminishing efficacy of conventional techniques, particularly signature-based ones. By means of its examination of the literature, the study aims to offer a thorough grasp of the current state of deep learning-based malware identification. It offers insight into the practical challenges faced in real-world implementations by examining diverse approaches, sophisticated frameworks, and dataset exploitation.

The continuous conflict between cybersecurity professionals and cybercriminals is exacerbated by the persistence of malware, which is created to cause harm and interfere with computer systems. This conflict is getting worse, which means that it is time for a paradigm shift as we move from static techniques to tactics that are flexible and robust. Deep learning is an effective technology that can automatically extract meaningful features from raw data, based on neural networks that are similar to those found in the human brain.

The two main tools in the fight against malware are recurrent neural networks (RNNs) and convolutional neural networks (CNNs). They are skilled at seeing small trends, which makes it possible to identify risks that were previously unknown, such as zero-day malware vulnerabilities. When it comes to navigating the ever-changing malware landscape, deep learning models' versatility is invaluable when compared to strict signature-based approaches.

However, there are difficulties in combining malware detection with deep learning. The robustness and stability of these models continue to be crucial issues, especially in the face of hostile attempts to trick systems. It's critical to maintain constant watchfulness against attacks that try to take advantage of weaknesses. Furthermore, overcoming interpretability problems in deep learning models poses formidable challenges, requiring a careful balance between transparency and precision.

By overcoming current obstacles, this study seeks to advance the continuing conversation in the field. The need to use deep learning is becoming more and more important given the growing amount of digital data available and the increasing severity of malware threats. This research is driven by the need to not only identify but also prevent malware attacks, as well as the promise that comes with combining deep learning with cybersecurity.

This commitment is the result of realizing the unique advantages of deep learning models, which are capable of autonomously extracting relevant features from raw data. In particular, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) show themselves to be powerful instruments, locating hidden vulnerabilities like zero-day malware threats and unveiling cryptic patterns. But overcoming these challenges necessitates a multipronged strategy for deep learning integration with malware detection.

Current approaches, potential innovations, and future paths contribute to the continuing conversation about malware protection. The goal of this project is to significantly advance current efforts to strengthen defenses against persistent attacks. In light of the constantly evolving cybersecurity landscape, the reasoning emphasizes the importance of pushing the boundaries of knowledge and innovation in order to achieve a safer digital environment amid the complex world of deep learning-driven malware detection.

### **1.3. Research Question**

"How can the integration of deep learning algorithms enhance the efficacy of malware detection, considering the evolving nature of cyber threats and the need for adaptive, proactive defense mechanisms in comparison to traditional signature-based approaches?"

### **1.4. Research Objectives**

- Analyze and evaluate the effectiveness of deep learning algorithms, particularly Graph Neural Networks (GNN), in automatically extracting relevant features from raw data in order to obtain accurate and quick malware identification.
- Compare the flexibility and adaptability of deep learning models to static signature-based techniques in order to gain an understanding of their capacity to identify and respond to new and unknown malware variations.
- Examine the stability and dependability of traditional deep learning algorithms and compare them with real-world graph-based deep learning methods. As part of this investigation, adversarial attack weaknesses are evaluated, and strategies to strengthen resilience against deceitful cybercriminals' tactics are developed.
- Examine the interpretability issues associated with deep learning models in the context of malware detection, with the goal of finding a compromise between accuracy and transparency for reliable decision-making processes.
- Examine whether deep learning-based malware detection systems are scalable, addressing any obstacles and optimizing model performance to handle the exponential expansion of digital data.
- Combine the results of the literature research to identify future directions and possible advancements in the field, providing a roadmap for advancing the integration of deep learning with cybersecurity to strengthen proactive and intelligent malware defense.

### **1.5. Research Outline:**

An overview of the goals and context of the study is given in this chapter. The report's subsequent sections are organized as follows:

- **Chapter 2. Literature Review:**

An extensive review of earlier studies on malware detection is provided in this section, with a focus on deep learning techniques. Key findings from multiple studies are summarized in a table, highlighting areas of current research need as well as future directions for investigation.

- **Chapter 3. Methodology:**

Systematic methods for gathering data, preparing it, and modeling it using deep learning techniques are described in detail in the methodology section. Providing details about each model's design, parameters, and training methods upholds transparency in the research process.

- **Chapter 4. Results and Discussion:**

The results of using deep learning models to identify malware are shown in this section, along with a performance metrics analysis and a discussion of new developments and difficulties. Within the context of malware identification, the debate offers a detailed assessment of these models' advantages and disadvantages.

- **Chapter 5. Conclusion and Future Scope:**

The research findings are outlined in the conclusion, with a focus on the contributions and implications for malware detection. The section on prospects for the future outlines directions for future research with the goal of promoting continuous improvements in the use of deep learning for efficient malware detection.

## Chapter 2: Literature Review

### 2.1 Concept of Malware Detection Using Deep Learning

Within the quickly changing field of cybersecurity, "malware" appears as a common and enduring threat that puts people, companies, and even whole countries at risk. Traditional malware detection techniques are unable to adequately address this threat as malicious software becomes more complex and digital data grows exponentially larger. Rather, new and cutting-edge tactics are essential. In this regard, there is great potential for identifying and blocking malware assaults at the junction of deep learning and cybersecurity (Awan et al., 2021). This review of the literature explores the complex integration of different fields, examining the subtle strategies and techniques used in this study. Researchers and industry experts have worked to create proactive malware detection systems that use advanced architectures and neural networks to detect and contain harmful software (Yuan et al., 2020).

By clarifying the many approaches, frameworks, and datasets used in deep learning-based malware detection, this literature review seeks to present a thorough examination of the state of the art in this field of study. The difficulties that arise in applying these models in the actual world are addressed, including interpretability, scalability, and adversarial attacks. In addition, by utilizing the knowledge and insights gained from recent studies (Wang et al., 2020) to clarify the current status of research, the review proposes future directions and possible developments in this important field.

The ongoing conflict between cybersecurity professionals and cybercriminals has intensified with the advent of malware, which is intended to compromise computer systems and wreak damage. The growing quantity and complexity of contemporary malware strains have revealed the waning effectiveness of conventional detection techniques, like signature-based methods (He and Kim, 2019). In response to this dynamic threat landscape, deep learning algorithms' incorporation into malware detection techniques represents a paradigm change. At the vanguard of this revolutionary approach is deep learning, a type of machine learning that draws inspiration from the neural networks found in the human brain. "Deep learning's"

capacity to independently identify pertinent features from unprocessed data is one of its main benefits when it comes to virus detection. These characteristics cover an extensive range of malware characteristics, from file types and API calls to execution-related behavioral patterns. More specifically, zero-day malware dangers and other threats that have not yet been identified can be easily recognized because to the exceptional ability of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to discern these subtle and even elusive patterns.

According to Schaukat et al. (2020), deep learning models' flexibility is essential in the ever-changing world of malware. Deep learning algorithms have the potential to constantly adjust to newly discovered malware strains that have eluded detection, in contrast to static signature-based techniques. According to Ucci et al. (2019), there are a few obstacles to overcome when combining deep learning with malware identification. Making sure deep learning models are robust and dependable is crucial, particularly when confronted with hostile attempts to trick these systems. Deep learning models also pose interpretability challenges that require careful balancing acts between openness and precision in research projects (John et al., 2020).

## **2.2 Literature Review**

Malware attacks are a growing security concern that complicate the use of current detection methods, as Vinayakumar et al. (2019) point out. Presently available techniques entail a careful examination of malware signatures and activity. Because it eliminates the need for lengthy feature engineering, machine learning—and deep learning in particular—holds promise. Real-time application is hampered by bias in recent studies due to a lack of training data. Using a variety of data splits to reduce dataset bias, this study compares deep learning and conventional machine learning algorithms (MLAs) in malware identification. Introducing a new method for image processing that improves zero-day virus detection through parameter optimization is a major contribution.

According to Senanayake et al. (2021), the increasing use of Android smartphones has resulted in a rise in malware attacks that employ strategies including espionage and credential theft. In order to solve this, machine learning (ML) techniques have shown promise in identifying these threats without the need for pre-established fingerprints. The

analysis evaluates the benefits, drawbacks, and potential improvements of 106 carefully chosen papers on machine learning-based Android malware detection. It also looks at how machine learning techniques may be used to find vulnerabilities in source code, which is important for improving security after an app is released. The publication's objective is to provide researchers with a thorough understanding of the Android security space and to direct future research efforts.

According to Qiu et al. (2020), deep learning (DL) has had a revolutionary effect on cybersecurity research, particularly with regard to the detection of Android malware, due to the large amount of data and constantly changing threats. This review explores the use of DL to tackle data protection and obfuscation issues associated with Android malware. With a focus on code semantics representation, it classifies research based on DL architectures such as FCN, CNN, RNN, DBN, AE, and hybrid models. The assessment also looks at issues like architectural choices and data quality, giving an overview of the state of the sector right now. It contributes to the advancements in this emerging field by providing insights into possible directions and chances for further investigation.

With the increasing use of Android smartphones, Mahindru and Sangal (2021) provide "MLDroid," a web-based framework designed specifically for Android malware detection. In order to detect malware in Android applications, "MLDroid" uses dynamically selected attributes for dynamic analysis. By utilizing a model that combines deep learning, feature subset selection, Y-MLP, nonlinear ensemble, decision tree, forest algorithms, and farthest first clustering, it is able to detect real-world app malware with an astounding 98.8% detection rate. Both user privacy and system integrity are successfully protected by this model. A dataset with more than 500,000 Android apps is used to train the framework.

A strong malware detection system (MDS) is essential, according to He et al. (2019), because malware threats are growing annually. The method of feature extraction and selection in conventional machine learning is often based on classical machine learning, which can be tedious and inaccurate. This paper investigates the efficiency of Convolutional Neural Networks (CNNs) in reducing the insertion of duplicate APIs, whereas conventional deep learning depends on Recurrent Neural Networks (RNNs). CNN's resilience to these kinds of insertions is investigated in this study. In order to manage different input sizes in their malware research, they used CNN with spatial pyramid pooling layers (SPP)

in the detection system, to turn infected files into images. The results indicate memory limitations associated with the implementation of SPP and show how effective grayscale imaging is in lowering the number of unnecessary API injections and improving detection precision.

Usman et al.'s (2021) analysis emphasizes how serious a problem data breaches and cybercrimes are in this age of increasingly networked electronics. A vital part of preventing such threats is identifying malicious Internet Protocol (IP) addresses prior to contact. Even though the IP reputation system has potential as a cybersecurity tactic, problems with resource consumption, false positives, and administrative expenses make existing approaches ineffective. A creative hybrid strategy that combines machine learning, data forensics, cyber threat intelligence, and dynamic malware analysis has been put out to address these issues. This system does IP reputation prediction and zero-day attack classification by using Decision Trees in large data forensics. Enhancing IP address detection's efficacy and efficiency in protecting cyber-physical systems is the goal of this complex approach. Lifespan, confidence, danger, and severity are all thoroughly assessed in the evaluation. A comparison of the recall and precision of machine learning algorithms demonstrates the enhanced security aspects of the reputation system. To determine how effective the entire system is, it is also tested against current engines.

According to Bae et al. (2020), it is imperative to distinguish ransomware from other types of malware to protect users' devices in the face of the spread of various versions. Although ransomware is similar to other malware, it is distinguished by unique features, such as how quickly it can carry out file-related operations in order to encrypt victim data. Customized defensive tactics are required since zero-day ransomware assaults are difficult to detect by traditional signature-based malware detection methods. In order to successfully differentiate ransomware from other malware types and innocuous files, this study suggests a way for its detection. Test findings show that the technique works well for separating malware from ransomware in situations where a mix of malicious and benign files are present.

Deep belief networks (DBNs) are utilized by Yuxin and Siyi (2019) to identify malware, which they represent using opcode sequences. In contrast to traditional shallow neural networks, DBNs can use unlabeled input to pre-train a multi-layer generative model, improving the representation of data features. In comparison to the three baseline models, DBNs perform better in terms of accurate malware identification, according to performance evaluations using k-nearest neighbor algorithms, decision trees, and support vector machines.



Moreover, DBNs demonstrate their capacity to model data structures and drastically minimize feature vector size by functioning as efficient autoencoders to extract executable feature vectors.

According to Ren et al. (2020), Android has been crucial to the quick development of the Internet of Things (IoT), but it has also brought about security flaws because of malware. The time and effort needed to create static or dynamic characteristics impedes the effectiveness of current machine learning-based solutions for detecting Android malware. In order to overcome this constraint, this research presents two comprehensive methods for Android malware detection that rely on deep learning. These models, which use the raw bytecodes from Android applications, perform better than existing methods, with identification accuracies of 93.4% and 95.8%, respectively. These techniques are especially good for Android IoT devices because they don't limit the amount of input files, don't require human feature engineering, and use less resources.

According to Roseline et al. (2020), the growing threat presented by malware is making it difficult for conventional detection techniques to identify it because of its increased sophistication and use of obfuscation techniques. This study discusses the challenge of classifying unidentified malware variants and determining distinctive traits for every one of them. It provides a novel approach that uses a powerful machine learning-based anti-malware system that visualizes malware as two-dimensional images. Compared to other approaches, our layered ensemble strategy with deep learning inspiration performs better and does not require complex parameter tweaks or backpropagation. Its thorough feature representation demonstrates its exceptional effectiveness in identifying novel and complex malware, as demonstrated by its excellent detection rates of 98.65%, 97.2%, and 97.43% across a variety of malware datasets.

According to Venkatraman et al. (2019), malicious software attacks have increased as a result of the Internet's quick growth and people's increasing reliance on complex apps. In a time of networked gadgets and massive data flows, the growing threat of malware presents a serious security risk. Current methods for detecting viruses seem to be overwhelmed by the complexity and size of today's data rich environment.

Thus, by combining deep learning and visualization techniques, this work presents a novel hybrid strategy that aims to improve malware detection. In order to effectively categorize malware by spotting suspicious system behavior, this method combines deep learning architectures with image-based approaches. Several similarity measures and deep learning models with cost considerations are used in the performance evaluation process. Tested on large malware datasets from both public and private sources, the scalability of the approach demonstrates great classifier accuracy.

Malware attacks have historically targeted traditional PCs, but as the Industrial Internet of Things (IIoT) expands its device deployment to monitor data flow over complicated networks, they increasingly constitute a concern, as explained by Naeem et al. (2020). Defending millions of IIoT users against such damaging attacks requires a high-level security plan. The computational complexity of current malware identification methods frequently presents a challenge. This study presents the MD-IIOT architecture, which is intended to detect malware assaults on IIoT systems, in order to overcome this difficulty. Comparative results show that this approach to malware analysis delivers higher prediction speed and detection accuracy compared to other machine learning and deep learning techniques by merging deep convolutional neural networks with color picture visualization.

According to Gibert et al. (2020), current machine learning research is necessary for successful malware detection because of the constantly changing malware landscape and the ongoing conflict between security analysts and malware authors. With an emphasis on deep learning, this review methodically provides an overview of machine learning techniques used to combat malware. It makes a contribution by outlining the difficulties present in traditional machine learning workflows and examining current developments in the field, especially with regard to deep learning methodologies. In addition, it delves into open questions and new directions for research. This study gives researchers a thorough grasp of malware detection in the context of this dynamic and ever-evolving threat ecosystem. It also provides new insights into potential breakthroughs and research areas.

Feng et al. (2020) provide "MobiTive," a powerful Android malware detection system, in response to the growing threat posed by malware obtained through unofficial markets and unaffiliated channels. MobiTive functions as on-device defence mechanism, providing responsive and real-time security directly on mobile devices, in contrast to server-side techniques. By using customized deep neural networks that are designed for mobile settings,

the technique gets beyond performance constraints related to memory, processing speed, and battery usage. This study thoroughly assesses a number of variables, such as feature extraction techniques, feature kinds, deep neural network performance, real-time detection capabilities, and flexibility to accommodate changing mobile device specifications. When this workable strategy is put into practice, user security and usability are greatly improved.

Hemalatha et al. (2021) have observed that the increase in malware cases presents a substantial security risk that outweighs the effectiveness of conventional defense systems. This research presents a novel approach to malware detection that combines visualization and deep learning methods. The technique uses a DenseNet model to classify malware binaries that are shown as two-dimensional images. Comprehensive tests on four reference datasets confirm its capacity to identify new malware samples with increased precision and decreased false positives while consuming the least amount of computational resources. The suggested method seems to be dependable and effective even when faced with obfuscation attacks.

Pektaş and Acarman (2020) claim that the prevalence of Android applications and their malicious variations necessitates the implementation of strong malware detection techniques in order to ensure mobile platform security. In order to accurately simulate application behavior, this study uses call graphs from APIs to record call sequences, branching, and call order. On the other hand, traditional graph matching methods for classification can be laborious, unreliable, and slow. In order to overcome this, the authors suggest a novel method that uses deep neural networks by embedding API call graphs into low-dimensional numerical vectors. In terms of malware classification, remarkable results are obtained: 98.86% accuracy, 98.65% F-measure, 98.47% recall, and 98.84% precision. The focus of the research is on improving network performance by adjusting parameters and using different embedding strategies.

Because mobile devices hold a lot of sensitive data, there are a lot of risks associated with their widespread use. According to study by Iadarola et al. (2021), malevolent actors constantly create increasingly complex programs to steal personal data from these devices by taking advantage of the shortcomings of the signature-based anti-malware solutions that are now in use. The method's usefulness is validated by the outcomes of the experiment,

with an average accuracy of 0.96 to 0.97 in identifying Android malware across six families. Importantly, the results offer useful insights into model predictions.

According to Shaukat et al. (2020), as the internet becomes a more essential part of everyday life in a globalized society, the likelihood of cybercrimes and threats increases. Cybercriminals are always coming up with new ways to get around security measures, which results in a variety of cyberthreats, including illegal online activity. Advanced and zero-day assaults are frequently difficult for conventional techniques to detect. Many machine learning algorithms have been developed to address this. This paper investigates the efficacy of three well-known machine learning techniques in detecting malware, spam, and intrusions: deep belief networks, decision trees, and support vector machines. Through the use of well-known and widely used datasets, the study carefully assesses how effectively these strategies work against the constantly changing cyber threat scenario.

According to Azeez et al. (2021), malware threats present a serious concern to user security in the digital era due to the possibility of data loss and malicious conduct. Because malware is always changing, it is necessary to investigate intelligent detection techniques, especially for new and unknown threats, as standard detection systems cannot keep up with the rapid advancement of malware. In an ensemble learning paradigm for malware detection, this paper suggests using stacked ensembles of fully connected, one-dimensional convolutional neural networks (CNNs) as foundation stage classifiers. Then, for end-stage classification, a review of fifteen machine learning classifiers is carried out. The findings of investigations conducted on a Windows Portable Executable (PE) malware dataset were used to create the experimental results, which were acquired utilizing an ensemble of seven neural networks and the Extra Trees classifier as the final-stage classifier.

Data security is still crucial in the big data and cloud computing era, as highlighted by Lee et al. (2019). Even with the latest developments, ransomware attacks continue to pose a serious risk and demand payment in order to unlock encrypted data. Current techniques of detection, whether behavior-based or file-based, have difficulty identifying ransomware that contains undisclosed malicious code, particularly in cloud services and backup systems. This work presents a novel method that uses entropy analysis to measure data uniformly in addition to machine learning for encrypted file classification. In the event of a system breach, original data can be still recovered from backup systems thanks to this method's effective identification of ransomware-infected files.

It performs better than conventional detection methods, showing high detection rates and low false positives and negatives.

Ma et al. (2019) have brought attention to the serious hazards associated with the spread of Android malware, which include privacy breaches and privilege escalation. The paper presents a thorough machine learning-driven approach for Android malware detection in order to counter this threat. To improve detection capabilities, it first builds a strong ensemble model and an application control flow graph to record API data.

Singh and Singh (2021) draw attention to the widespread danger that malicious software, or malware, poses to consumers' online security. These concerns include the possibility of data loss or uninvited tampering. Traditional malware detection techniques frequently break, particularly when faced with unknown software. Their work suggests a communal learning strategy to address this issue. Base-stage classification is performed using stacked ensembles of fully connected, one-dimensional convolutional neural networks (CNNs), while end-stage classification is performed using a machine learning method. In order to determine the effectiveness of fifteen machine learning classifiers in combating malware threats, the research thoroughly examines these models, which include naïve Bayes, decision trees, random forests, gradient boosting, and AdaBoosting.

Zhong and Gu (2019) have emphasized the growing threat posed by sophisticated malware, which emphasizes the critical role deep learning-based malware detection systems (MDSs) play in protecting economic interests and national security. Although traditional techniques sometimes depend on a solitary deep learning model that has been trained on extensive datasets, the intricacy and variety of malware data distributions can be beyond the capabilities of current technologies. The Multi-Level Deep Learning System (MLDLS) was introduced with the goal of improving MDS performance in order to tackle this difficulty. MLDLS uses a tree structure to arrange several deep learning models, each of which is entrusted with learning distinct data distributions that correlate to various malware families. The use of obfuscation techniques by attackers frequently renders traditional machine learning models ineffective in the face of robust Android malware, as noted by Lee et al. (2019). An innovative method is put forth to overcome this difficulty, utilizing layered Convolutional and Recurrent Neural Networks (RNNs and CNNs) for more accurate malware categorization. CNN units, embedding, and

Gated Recurrent Units (GRU) are utilized to extract machine learning features, yielding better performance than Ngram-based models.

The current signature-based techniques used in anti-malware systems, according to Mercado et al. (2020), are essentially useless against malware that has been obfuscated. The research presents a sophisticated method for identifying malware families and identifying viruses on mobile devices. A multitude of classifiers are built upon the grayscale images that are produced by extracting various attributes from executable samples. As for the ransomware assaults, Poudya et al. (2019) draw attention to the significant losses that have been sustained in a number of industries, highlighting the necessity for sophisticated detection techniques. Their research examines ransomware using a multi-tiered big data mining strategy that combines machine learning, natural language processing, and reverse engineering techniques. At several levels, including the assembly instruction, function call, and dynamic link library levels, the framework uses supervised machine learning algorithms. A hybrid model that combines convolutional neural network (CNN) and deep autoencoder (DAE) architectures is suggested in order to improve the accuracy and effectiveness of Android malware detection on a large scale. The approach proposed by Wang et al. (2019) is supported by their emphasis on improving feature extraction through the restoration of high-dimensional features of Android apps. Furthermore, a serial CNN architecture (CNN-S) is applied, which incorporates dropout mechanisms and non-linear activation functions to reduce overfitting. CNN uses deep autoencoder as a pre-training technique in order to speed up training procedures.

In their research, Pei et al. (2020) present "AMalNet," a novel deep learning framework designed to improve Android malware detection and family identification in the context of growing threats against Android and IoT devices. Independently Recurrent Neural Networks (IndRNN) are utilized by "AMalNet" to extract deep semantic information while taking into account distant dependencies among data points. Sequential patterns and high-level graphical semantics are captured by GCNs, while various embedding representations are used by IndRNN.

## 2.3 Table of Summary

Paper Name	Paper Author & Year	Paper Summary	Limitations
A hybrid deep learning image-based analysis for effective malware detection	Venkatraman, S., Alazab, M. and Vinayakumar, R., 2019	This study compares deep learning and conventional MLAs for malware detection while avoiding dataset bias, as malware assaults are becoming an increasingly common security issue. It offers a novel method of image processing for zero-day vulnerability detection.	Bias brought about by a lack of training data and a lack of information regarding the new image processing technique.
Android mobile malware detection using machine learning: A systematic review	Senanayake, J., Kalutarage, H. and Al-Kadri, M.O., 2021	Potential improvements, source code vulnerabilities, and machine learning-based Android malware detection are evaluated in this study. It aims to direct next research projects and provide academics with a thorough understanding of Android security.	Restricted knowledge of certain methods and outcomes; the quantity of articles could not accurately reflect the field.
A survey of android malware detection with deep neural models. ACM Computing Surveys	Qiu, J., Zhang, J., Luo, W., Pan, L., Nepal, S. and Xiang, Y., 2020	The work investigates the use of deep learning in Android malware detection with a focus on multiple DL architectures. It provides insights for future study by discussing data quality, design choices, and the state of the field.	Lacks comparisons and particular results between various DL designs.

<p>MLDroid— framework for Android malware detection using machine learning techniques.</p> <p>Neural Computing and Applications</p>	<p>Mahindru, A. and Sangal, A.L., 2021</p>	<p>This paper presents MLDroid, a web-based tool for Android malware detection. Through the use of deep learning and additional algorithms, it obtains a high detection rate. 500,000+ Android app dataset was used for training.</p>	<p>More specifics regarding the algorithms' functionality could be provided.</p>
<p>Malware detection with malware images using deep learning techniques</p>	<p>He, K. and Kim, D.S., 2019</p>	<p>This research examines the application of spatial pyramid pooling layers in convolutional neural networks for malware detection. Grayscale imaging is examined and the problem of API injection is addressed.</p>	<p>Minimal consideration of real-world relevance and practical implementation issues.</p>
<p>Intelligent dynamic malware detection using machine learning in IP reputation for forensics data analytics</p>	<p>Usman, N., Usman, S., Khan, F., Jan, M.A., Sajid, A., Alazab, M. and Watters, P., 2021.</p>	<p>By combining data forensics, machine learning, and dynamic malware analysis, this study suggests a hybrid method for IP address reputation systems. The effectiveness and security elements of the system are assessed.</p>	<p>Limited information about the precise methods and algorithms employed, possible bias in threat intelligence and data forensics.</p>



Ransomware detection using machine learning algorithms. Concurrency and Computation: Practice and Experience,	Bae, S.I., Lee, G.B. and Im, E.G., 2020	The goal of the study is to differentiate ransomware from other forms of malware. It provides an approach to identify ransomware and distinguish it from other types of files. Test results show that the approach works well with heterogeneous datasets.	There are no explicit details given regarding the limits of the procedure.
Malware detection based on deep learning algorithm. Neural Computing and Applications	Yuxin, D. and Siyi, Z., 2019	In this paper, malware encoded as opcode sequences is identified using deep belief networks (DBNs). When it comes to accurate malware identification, DBNs perform better than baseline models, especially when given more unlabeled data for pretraining.	Information on possible downsides and real-world restrictions is absent from the study.
End-to-end malware detection for android IoT devices using deep learning	Ren, Z., Wu, H., Ning, Q., Hussain, I. and Chen, B., 2020	With excellent identification accuracy, the study offers end-to-end deep learning-based techniques for Android malware detection. These models don't have any input file size limitations, use raw bytecodes, and don't require human feature engineering.	It is not addressed what particular difficulties and restrictions arise while using these techniques.

Intelligent vision-based malware detection and classification using deep random forest paradigm	Roseline, S.A., Geetha, S., Kadry, S. and Nam, Y., 2020	In this research, a machine learning-based antimalware system that visualizes malware as two-dimensional images is presented. The technology performs better than rivals in identifying novel and complex malware without the need for parameter tweaks or backpropagation.	There are no specifics about the method's actual difficulties or restrictions given.
Robust intelligent malware detection using deep learning	Venkatraman, S., Alazab, M. and Vinayakumar, R., 2019	The goal of the project is to enhance malware detection with a hybrid methodology that combines deep learning and visualization. It makes use of deep learning architectures that combine several similarity metrics with image-based methods.	The limitations and difficulties in putting the hybrid methodology into practice are not mentioned in the paper.
Malware detection in industrial internet of things based on hybrid image visualization and deep learning model	Naeem, H., Ullah, F., Naeem, M.R., Khalid, S., Vasan, D., Jabbar, S. and Saeed, S., 2020	The paper proposes the "MDIIOT" architecture for malware analysis and discusses malware assaults on the Industrial Internet of Things (IIoT). It blends the display of color images with deep convolutional neural networks.	It doesn't address any specific restrictions or difficulties with MD-IIoT implementation.

*Table 2. 1: Table of Summary*

### Chapter 3: Methodology

A commonly used framework for directing all aspects of data mining and machine learning projects is called CRISP-DM (Cross-Industry Standard Process for Data Mining). The methodical development of deep learning-based malware detection solutions is made easier by its organized methodology, which also guarantees a methodical navigation through the challenges of comprehending business needs, preparing data, creating models, and implementing them. As noted by Podder et al. (2023), applying CRISP-DM to malware detection guarantees a systematic and well-documented process for building robust and effective deep learning models.

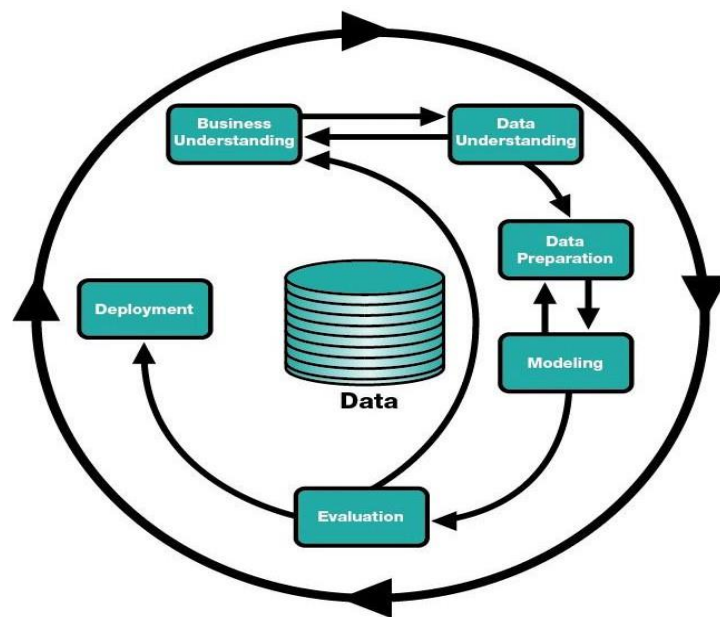
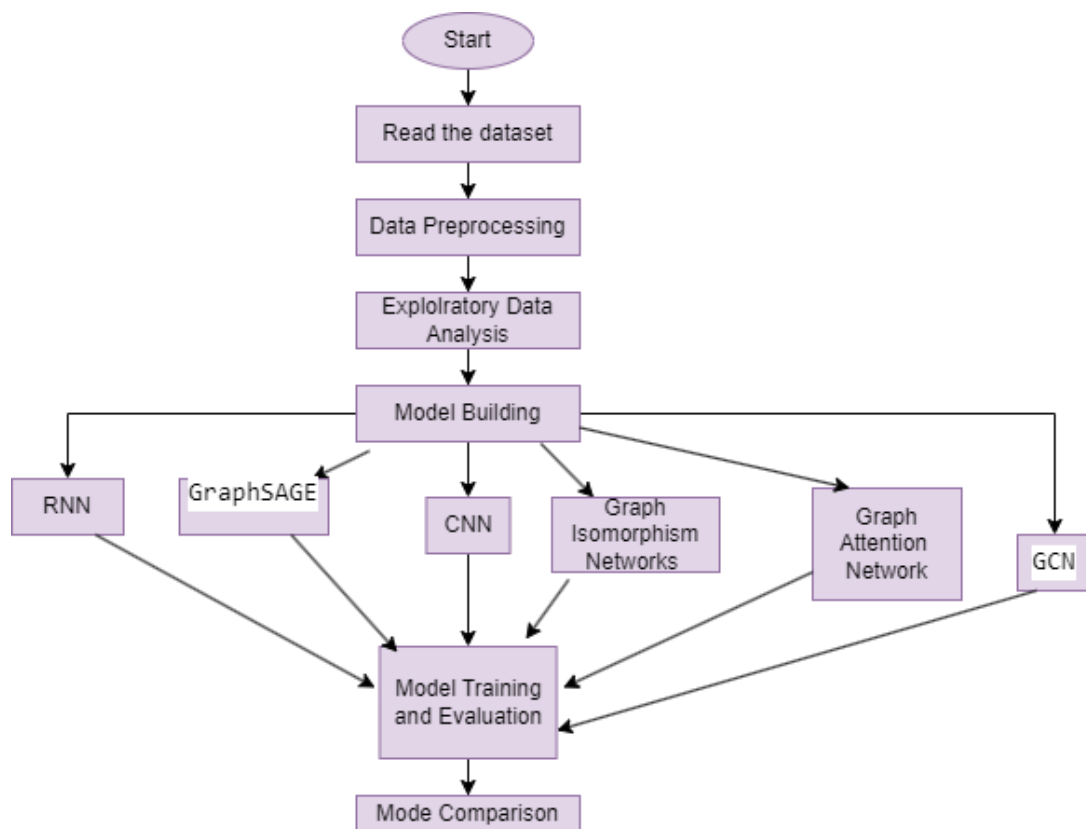


Figure 3. 1: CRISP-DM

The steps involved in CRISP-DM are:

1. **Business Understanding:** Identify and remove superfluous columns, including "Name," "Machine," and "TimeDateStamp," to efficiently simplify the dataset and show that you understand the data and the objectives of the firm.
2. **Data Understanding:** To learn more about how characteristics are organized and how important they are for malware identification, divide the dataset into independent variables (X) and dependent variables (y).

3. **Data Preparation:** To ensure consistency in scaling and data preparedness for training the model, separate the data into training and testing sets and standardize the characteristics with tools like StandardScaler.
4. **Modeling:** In order to mitigate the prevalent class imbalance in malware detection, wherein malware cases are frequently underrepresented, modeling techniques such as the Synthetic Minority Over-sampling Technique (SMOTE) should be utilized.
5. **Evaluation and Deployment:** The model is trained on the training set and assessed on the testing set to determine its performance during this implicit phase. To assess the efficacy of the model, evaluation criteria such as accuracy, precision, recall, and F1 score are crucial.



*Figure 3. 2: Project Methodology*

**Project Requirement:** We used Google Colab as our development platform in order to adhere to project standards. The main reason we choose it was because of its easy-to-use interface, which is akin to Anaconda Navigator and allows code running without the need for installation. Its file-sharing features also make cooperation and project management easier, which improves collaborative effectiveness inside the project.

**Python:** This programming language is well-known for its large library and versatility. It is especially well-liked in machine learning applications because of its high level nature and flexibility.

**TensorFlow:** Because of its versatility and scalability in creating and refining deep learning models, TensorFlow is a highly recommended open-source machine learning framework. Google's development activities were the source of it.

**Scikit-learn:** For a variety of applications including data analysis, clustering, regression, and classification, the flexible Python machine learning toolkit Scikit-learn (sklearn) offers easy-to-use capabilities.

**Matplotlib:** A large number of static, interactive, and excellent data visualizations are created for analysis and presentation using Matplotlib, a feature-rich Python charting package.

**Seaborn:** A neat and user-friendly interface for creating educational statistics visualizations is provided by the statistical data visualization library Seaborn.sophisticated.

**Deep learning models:** Graph Isomorphism Network, Graph Attention Network, RNN, GraphSAGE, and GCN were all used in this investigation.

### 3.1 DataSet:

I used a dataset from Kaggle for my research and the link is <https://www.kaggle.com/datasets/amauricio/pe-files-malwares>. 19,611 rows and 79 columns make up the dataset. E\_magic, e\_cblp, e\_cp, e\_crlc, e\_cparhdr, e\_minalloc, e\_maxalloc, e\_ss, e\_sp, SectionMaxChar, SectionMainChar, DirectoryEntryImport, DirectoryEntryImportSize, DirectoryEntryExport, ImageDirectoryEntryExport, ImageDirectoryEntryImport, ImageDirectoryEntryResource, and ImageDirectoryEntrySecurity are some of the features that are included in the column. The collection has float64, int64, and object data types, and it contains no null values. Malware is the variable of interest.

The data was imported as a DataFrame once our CSV file was loaded using the pandas read function. Later on, we cleaned the data and checked for null values. In order to transform categorical data into numerical format, label encoding was applied. The data was then plotted using the Matplotlib, with the Malware Target Variable, to highlight different features of the data.

Then, to guarantee consistent distribution among features, standardization was implemented. The pertinent information will be briefly covered in each of the ensuing subsections. An overview of the dataset is given here, along with a description of the preprocessing, visualization, and analysis procedures that were carried out.

### 3.2 Data Analysis using Tableau

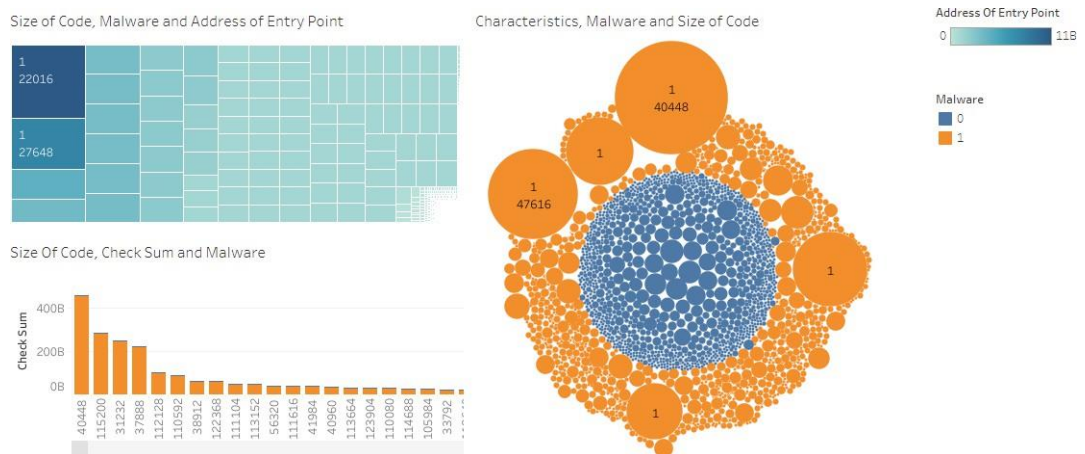


Figure 3. 3: Dashboard 1

A treemap showing the extent of code, instances of malware, and entry point addresses can be found in the first worksheet of the dashboard that comes before. Comparing darker to lighter areas, the former show higher concentrations of entry sites. A bubble chart illustrating traits, malware instances, and code size is presented in the second worksheet. Orange (1) and blue (0) are the malware instance colors. And lastly, a bar chart showing code size, checksum values, and virus incidences may be found in the third worksheet. Malware occurrences are shown by orange (1) and blue (0) colors, while code size is shown on the x-axis and checksum on the y-axis.

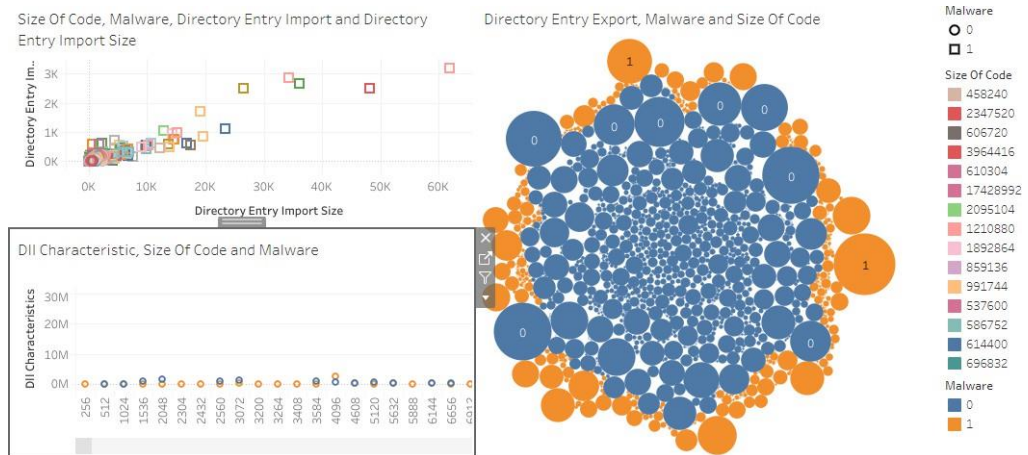


Figure 3. 4: Tableau Dashboard 2

The first worksheet in the dashboard above displays a scatterplot that shows the size of code, instances of malware, Directory Entry Import, and Directory Entry Import Size. various code sizes can be distinguished by various colors. A bubble chart showing Directory Entry Export, malware occurrences, and code size is presented in the second worksheet. Malware is represented by 1 or orange, whereas benign occurrences are shown by 0 or blue. Finally, a circular view demonstrating Dll Characteristic, Code Size, and Malware is displayed in the third worksheet. Variations in code sizes are indicated by different colors.

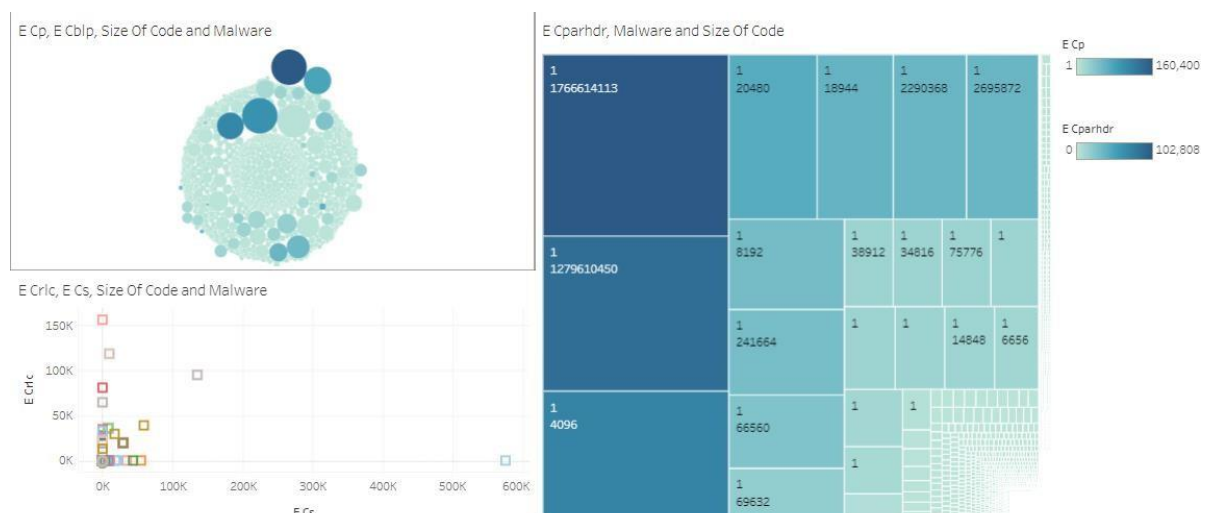


Figure 3. 5: Tableau Dashboard 3

A bubble chart displaying E Cp, E Cblp, Size Of Code, and Malware is displayed in the first worksheet of the dashboard that is supplied. Greater numbers of E Cp are found in darker areas than in lighter ones. The malware, size of code, and E Cparhdr are represented by a treemap in the second worksheet. Greater numbers of E Cparhdr are shown by darker regions. In the third worksheet, E Crlc, E Cs, Size Of Code, and Malware are shown in a scatterplot. With E Crlc on the y-axis and E Cs on the x-axis, different colors correspond to different code sizes.

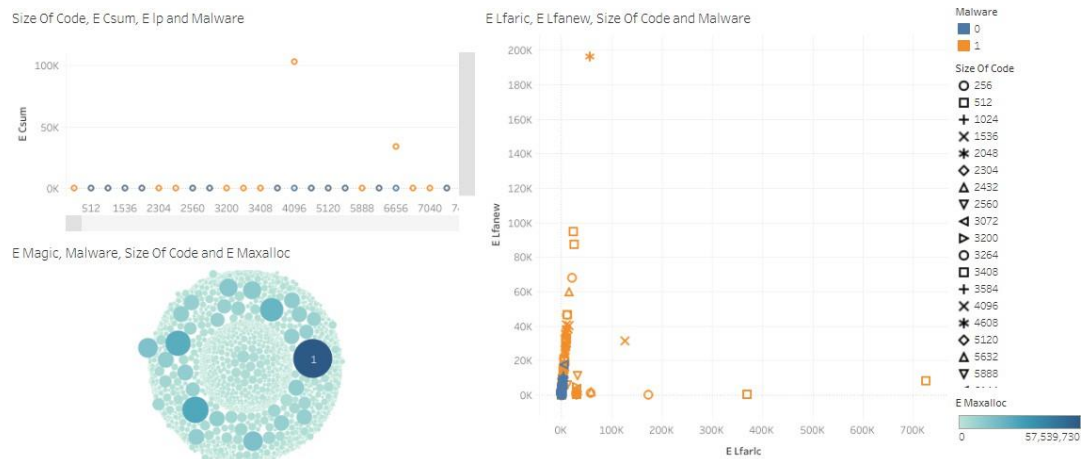


Figure 3. 6: Tableau Dashboard 4

Size Of Code, E Csum, E Ip, and Malware are represented by a circular view in the first worksheet of the dashboard that is provided. Malware occurrences are shown in orange (1) while benign ones are shown in blue (0) in this example. The second worksheet presents a scatterplot for the following variables: malware, size of code, elfarc, and elnew. Distinct symbols are used to denote different code sizes. The final worksheet has a bubble chart that shows E Magic, E Maxalloc, Size Of Code, and Malware. In comparison to brighter parts, darker regions show a larger count of E Maxalloc.



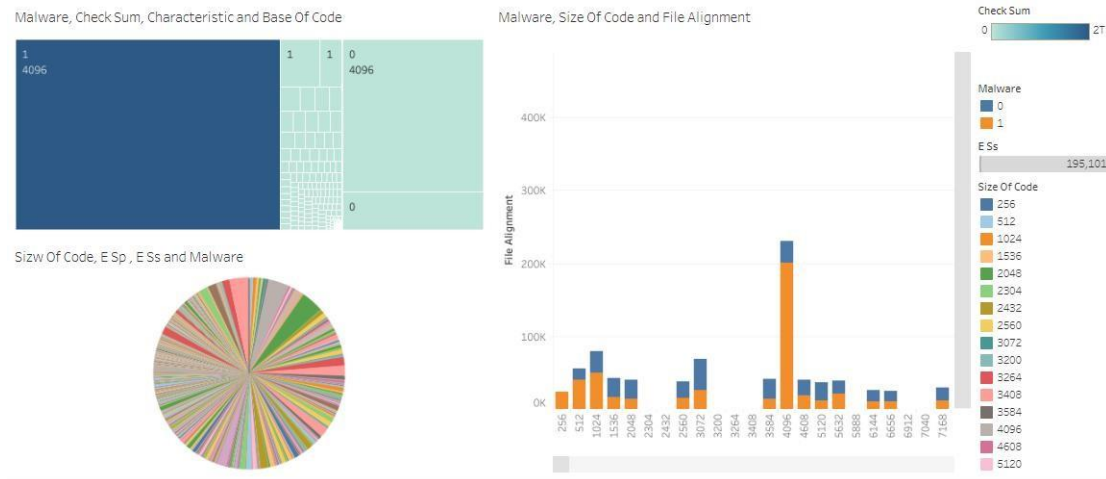


Figure 3. 7: Tableau Dashboard 5

A treemap depicting malware, check sum, characteristic, and base of code is shown in the first worksheet in the attached figure. In comparison to brighter zones, darker regions show a higher count of check sum. A stacked bar plot for malware, code size, and file alignment is shown in the second worksheet. Malware cases are depicted in orange, whereas benign instances are shown in blue. The final worksheet concludes with a pie chart that displays the Size of Code, E Sp, E Ss, and Malware. Code of different sizes is represented by different colors.

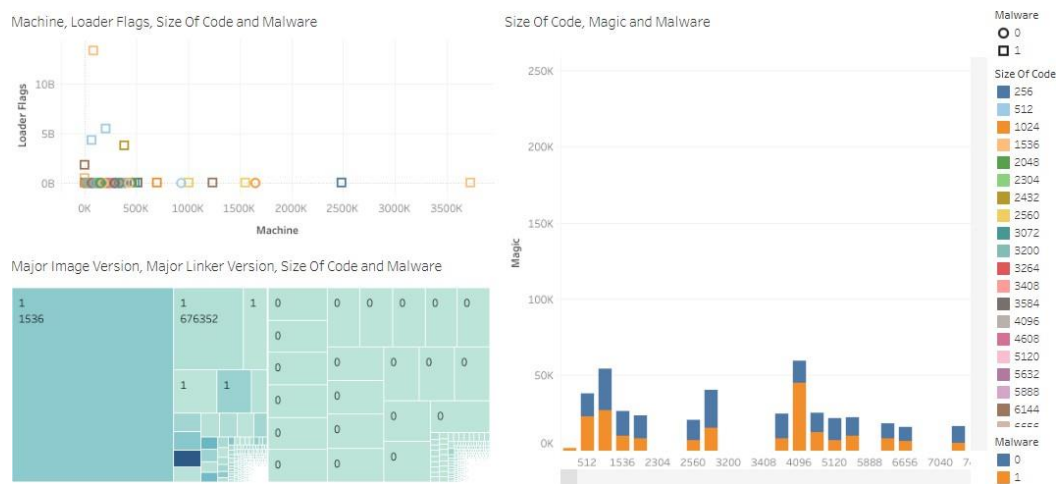


Figure 3. 8: Tableau Dashboard 6

The worksheet with machine, loader flags, code size, and malware is plotted in the dashboard above. The x-axis displays machine values, and the y-axis displays loader flags. The use of different colors to denote different code sizes is employed. Malware, magic, and code size are all shown in a stacked bar chart in the second worksheet. Malicious examples are shown in orange, and benign examples are shown in blue. The stack bar for size of code, magic, and malware is displayed in the second worksheet; in this case, blue indicates benign code and orange indicates malware.

Image Version, Major Linker Version, Code Size, and Malware, here the dark areas in this diagram represent the highest number of major linker versions relative to the bright areas.

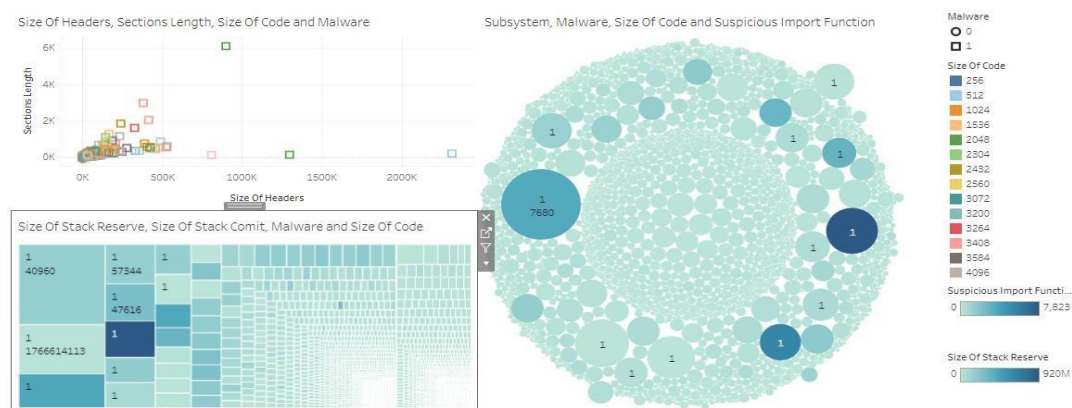


Figure 3.9: Tableau Dashboard 7

The first spreadsheet in the dashboard above shows a scatterplot that shows the following: Header Size, Section Length, Code Size, and Malware. The x-axis displays the header size, and the y-axis shows the section length. To indicate different code sizes, different colors are used. A bubble chart displaying Subsystem, Malware, Size Of Code, and Suspicious Import Function is included in the second worksheet. Darker areas indicate more suspicious import functions than lighter ones. The size of code, malware, size of stack reserve, and size of stack commit are represented by a tree map in the third worksheet. The darker areas indicate a higher number of size of stack reserve than the lighter ones.

### 3.3 Data Preparation

To ensure the model's efficacy and generalizability to previously unknown data, raw data for machine learning must be cleaned and formatted. The procedure comprises dividing the dataset into training and testing sets, scaling features, encoding categorical variables, and handling missing values. For further modeling activities, it is imperative to improve the quality and consistency of the input data.

```

#Import all needed libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import tensorflow as tf
import torch.nn as nn
import dgl.function as fn
from dgl import DGLGraph
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Input, Dropout, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
import torch.nn.functional as F
import torch
from tensorflow.keras.layers import Conv1D, GlobalMaxPooling1D
from tensorflow.keras.losses import sparse_categorical_crossentropy
from tensorflow.keras.metrics import sparse_categorical_accuracy
from sklearn.model_selection import StratifiedKFold
from imblearn.over_sampling import SMOTE
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

```

*Figure 3. 10: Import all libraries*

Essential libraries for deep learning, machine learning, graph neural networks, and data manipulation are imported by the given code. Seaborn, DGL (Deep Graph Library), TensorFlow, scikit-learn, NumPy, and PyTorch are notable libraries for in-depth data analysis and modeling. To improve code readability, warning messages are also filtered.

```

#Read the dataset
df_malware = pd.read_csv('dataset_malwares.csv')
df_malware.head()

```

*Figure 3. 11: Read the dataset*

Panda's read\_csv function is used to import the data from a CSV file into a DataFrame. For clarity and to check that the import was successful, the first five rows can be displayed using the head() method to provide a brief summary of the structure and content of the dataset.

To find null values in the dataset (df\_malware), the code uses the isnull().sum() method. This brief method provides a rapid summary of any data discrepancies by quickly identifying and tallying missing values across all columns.

During additional analysis or model training, it could be necessary to handle these null values in order to ensure data integrity.

```
#Checking the null values  
df_malware.isnull().sum()
```

*Figure 3. 12: Checking the null values*

The code identifies null values in the dataset (df\_malware) by using the isnull().sum() method, which essentially summarizes the missing data in every column. It might be essential to address these null values in later analysis or model training in order to guarantee data integrity and reliable outcomes. For the analysis or model to continue to be reliable and useful, this phase is essential.

```
#Statistical description of dataset  
df_malware.describe()
```

*Figure 3. 13: Statistical Description of Dataset*

Using the df\_malware dataset, the supplied code snippet uses the describe() method to provide statistical summaries. For each numerical column, it displays the mean, standard deviation, minimum, 25th percentile, median (50th percentile), 75th percentile, and maximum values, among other crucial information. This succinct synopsis makes it easier to understand the numerical properties of the dataset, such as its variability and distribution.

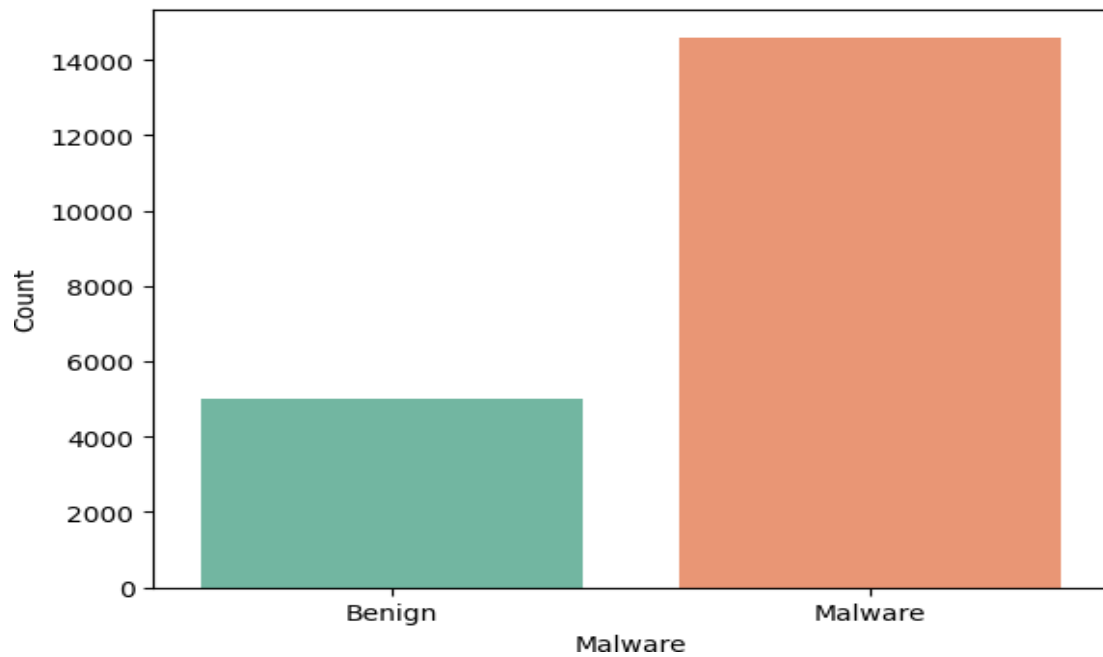
```
#Drop the unwanted columns from the dataset  
df_malware_f = df_malware.drop(['Name', 'Machine', 'TimeStamp'], axis=1)
```

*Figure 3. 14: Drop the unwanted columns*

I have removed unnecessary columns from the dataset, such as Name, Machine, and TimeDateStamp, in the aforementioned graphs.

### **3.4 Data Visualization:**

In order to clarify trends, patterns, and insights, data visualization is the process of presenting data graphically. To improve understanding and support decision-making in a variety of fields, including analytics, science, and business, it makes use of graphs, charts, and other visual aids.



*Figure 3. 15: Count plot for Malware*

A count plot is a form of categorical plot used in data visualization that shows the frequency of each category in a dataset. It is a straightforward method that works well for showing how categorical variables are distributed. Malware is shown as a count plot in the figure, where it is most frequently found in comparison to benign cases.

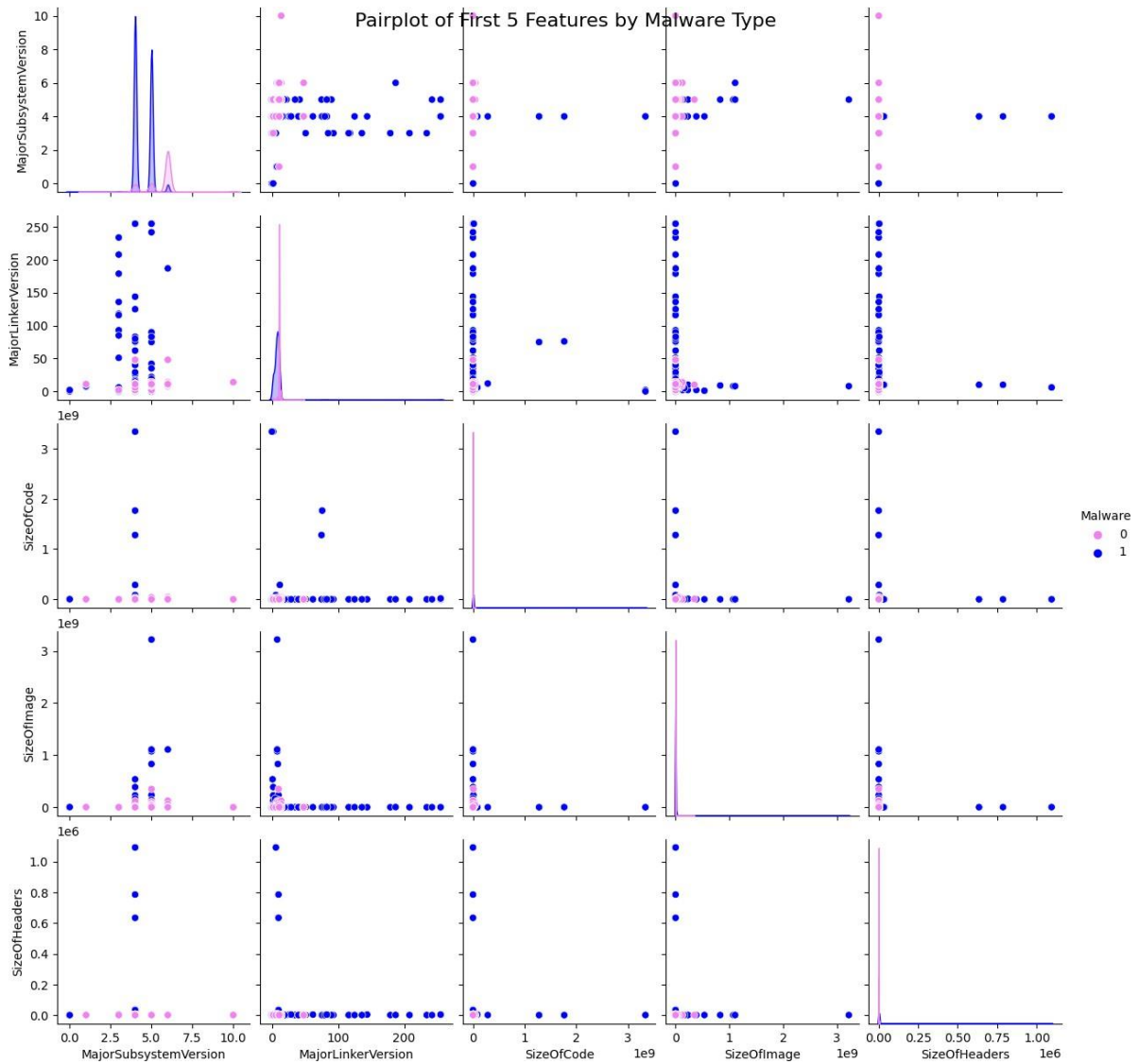


Figure 3. 16: Pair plot for first 5 columns

Pairwise connections within a dataset are displayed using a pair plot, a visualization tool in Seaborn. With the use of scatterplots for numerical variables and histograms for univariate distributions, it briefly describes variable relationships and distributions. In addition to helping with pattern recognition and malware and non-malware instance separation, the pair plot that is produced provides a thorough understanding of feature relationships.

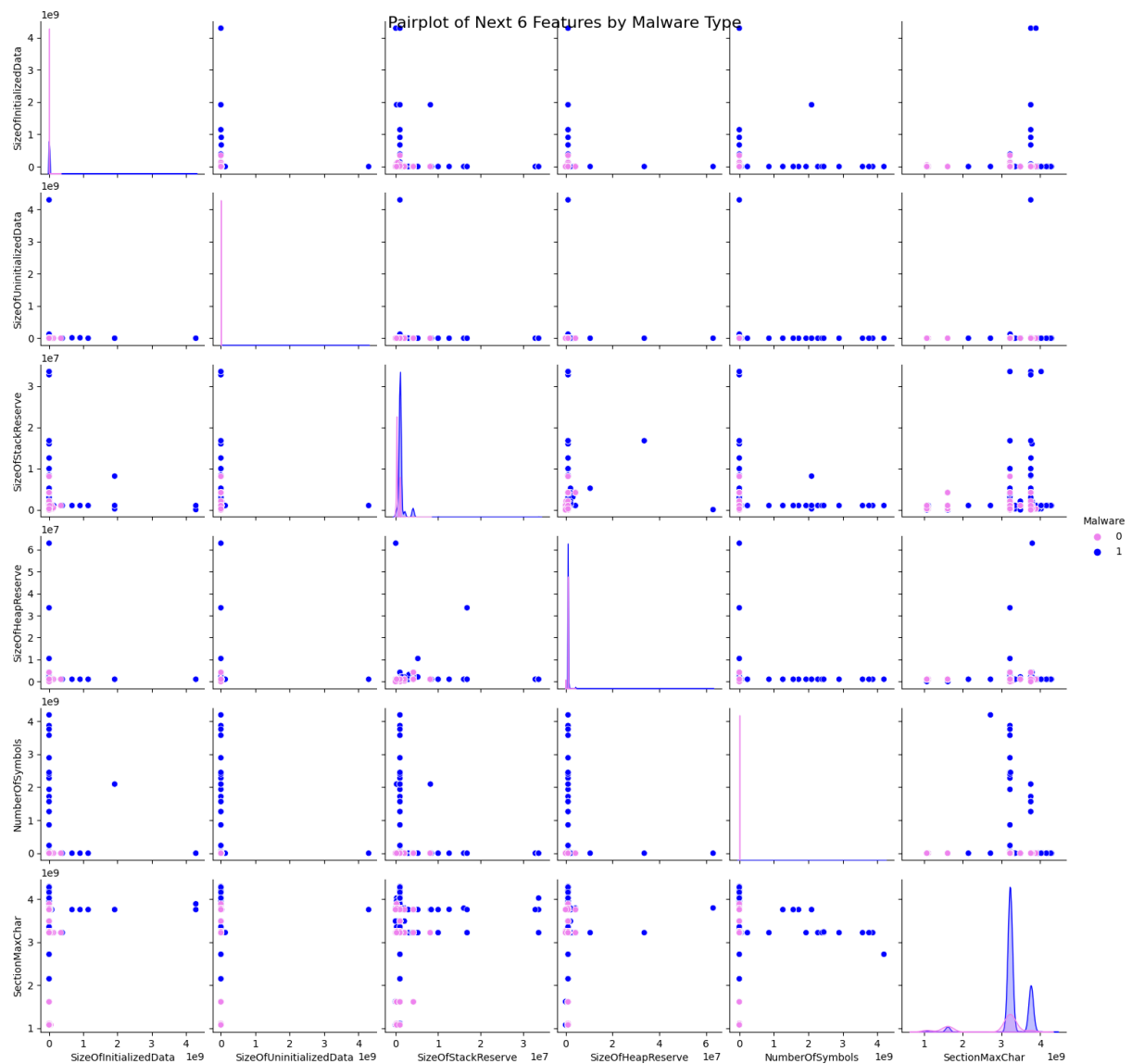


Figure 3. 17: Pair plot for next 6 columns

The pair plot in the DataFrame `df_malware_f` shows scatterplots and histograms for the next six features, which are color-coded according to the 'Malware' column. With the use of this visualization, patterns and distributions can be found, providing information about possible differences between malware and non-malware cases.



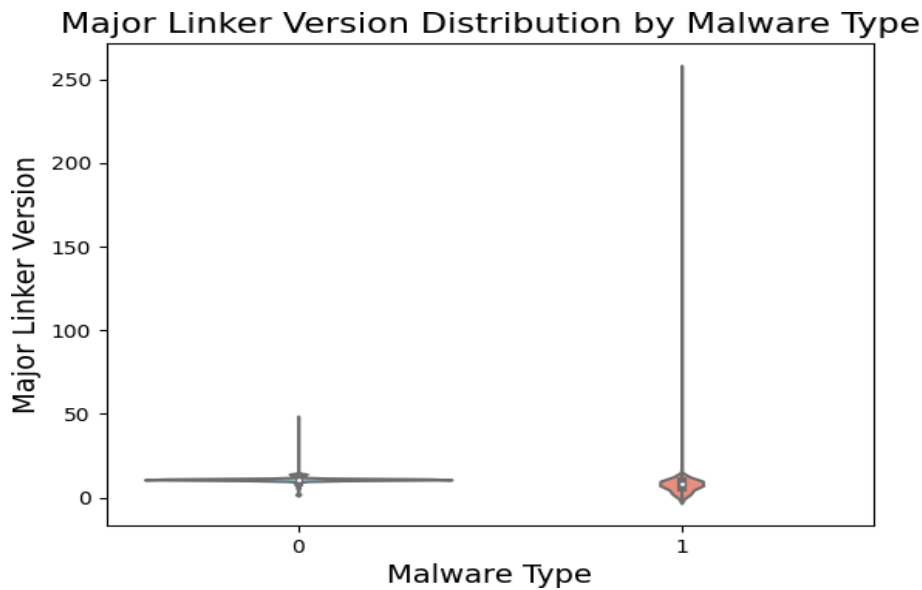


Figure 3. 18: Violin Plot for Malware Type and Major Linker Version

The distribution of the 'MajorLinkerVersion' feature across various 'Malware' categories is indicated by the Violin figure, which was created using Seaborn. The vertical violin represents each type of malware, and the width of the plot indicates the density of data points at different values of 'MajorLinkerVersion'. 'Skyblue' for non-malware and 'salmon' for malware are the two categories that are distinguished by the color scheme, which also offers information about the distribution and concentration of 'MajorLinkerVersion' values for each type of malware.

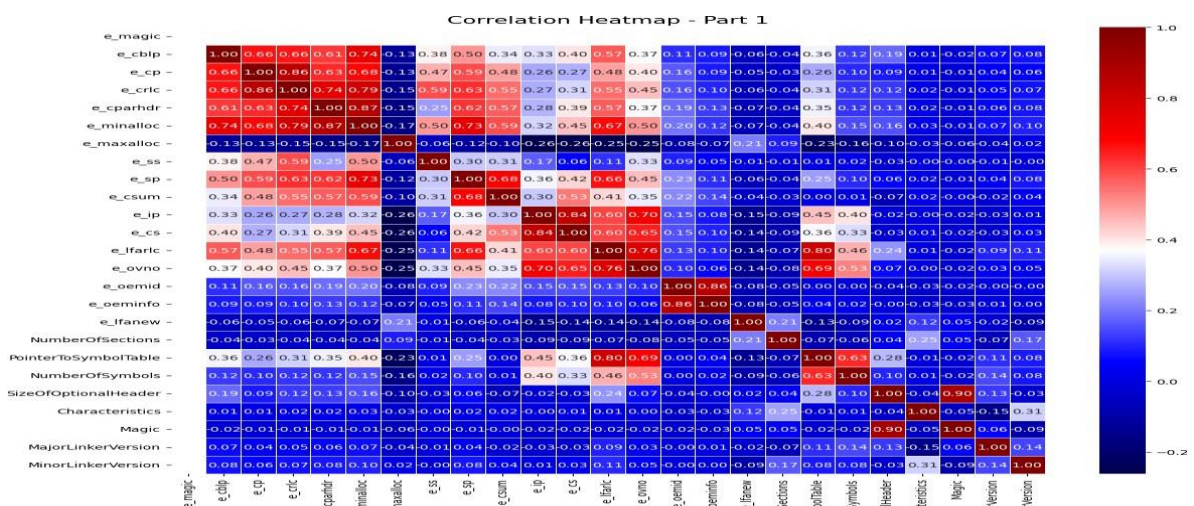


Figure 3. 19: Correlation Heat map 1



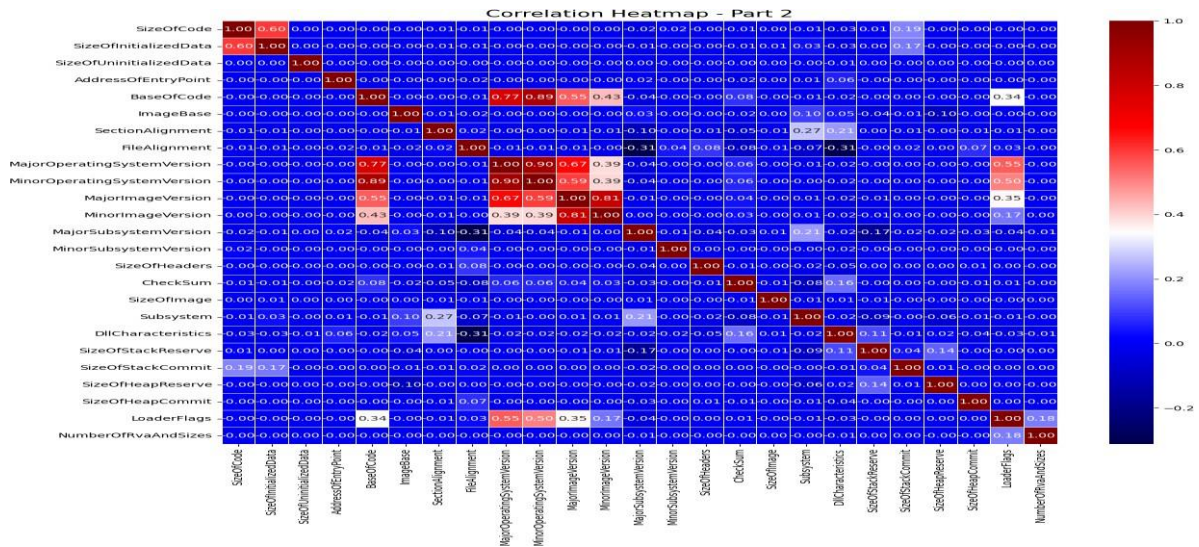


Figure 3. 20: Correlation Heat map 2

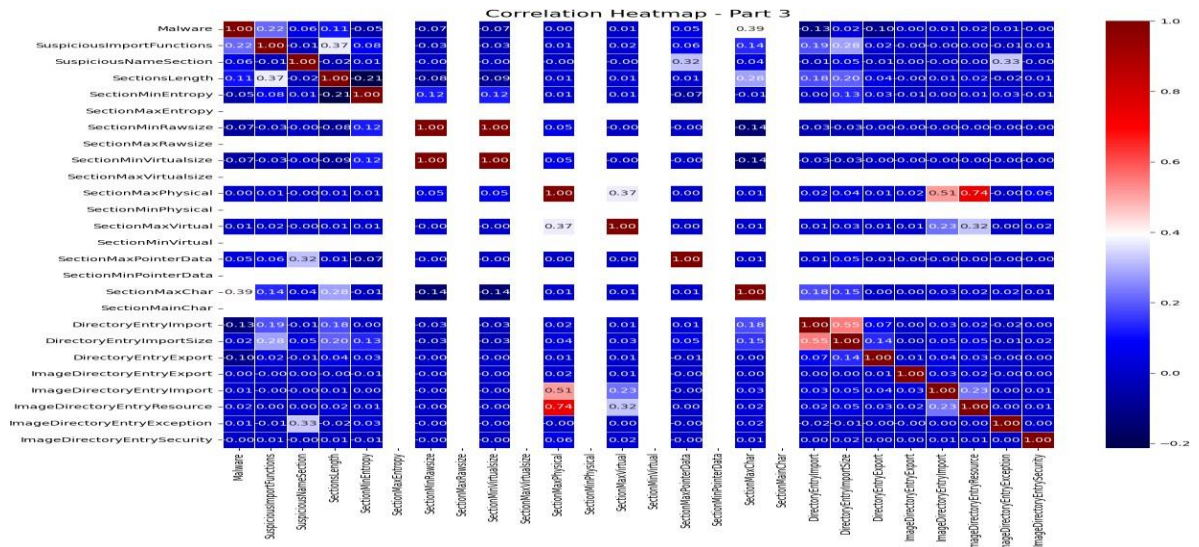


Figure 3. 21: Correlation Heat map 3

Using color to transmit information, a heat map is a visual depiction of values within a matrix. It provides information on the direction and intensity of interactions between the variables in correlation matrices. Generally speaking, cooler hues like blue denote negative correlations while warmer hues like red represent positive correlations. The correlation coefficient's magnitude can be seen in the color's intensity. Seaborn is used to create distinct correlation matrices for every part, which lead to matching correlation heat maps. By showing pairwise correlations between characteristics inside each area and providing annotated correlation coefficients, these heat maps aid in the discovery of patterns and relationships within the data.

### 3.5 Data Modelling:

The goal of data modeling is to use mathematical representations, or frameworks, to understand and analyze interactions within a dataset. This process, which is mainly related to machine learning, entails training and assessing algorithms to forecast or extract insights from data. The goal is to create models that show effectiveness when used with new, untested data so that they can be used for a variety of tasks like regression, classification, and clustering.

```
#Separate the dependent and independent variables  
X = df_malware_f.drop(['Malware'] , axis = 1)  
y = df_malware_f['Malware']
```

*Figure 3. 22: Separating the Dependent and independent variable*

The dataset df\_malware\_f is divided into dependent (y) and independent (X) variables using the code that is supplied. With the exception of "Malware," every column in X acts as an independent variable and adds characteristics to the predictions. The goal variable for classification or prediction is shown by the 'Malware' column in the dependent variable, y. This separation is a common procedure that is necessary to prepare data for machine learning, which makes it easier to train and assess predictive models.

```
# Split the data into training and testing sets  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

*Figure 3. 23: Perform train and test split*

The train\_test\_split function from the scikit-learn library is used by the code to divide the dataset into subgroups for testing and training. The goal variable and characteristics for training are contained in the resulting sets, X\_train and y\_train, whereas the corresponding sets for testing are represented by X\_test and y\_test. The data for the test set is allocated using the test\_size parameter, which is set at 20%. Reproducibility is ensured by using a fixed random seed by setting a random state. This section makes it easier to train and assess machine learning models on a variety of datasets.

```
# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

*Figure 3. 24: Normalize the dataset*

The algorithm first applies the same transformation to the test set (X\_test) after standardizing the numerical features in the training set (X\_train) using scikit-learn's StandardScaler. By guaranteeing that features have a mean of 0 and a standard deviation of 1, standardization encourages uniform scaling among variables. This preprocessing phase is critical to improving the efficiency of the model, especially for methods that depend on distance calculations or gradient-based optimization.

### **3.6 Model Used:**

The models that I have utilized are as follows:

#### **3.6.1. Convolutional Neural Network (CNN):**

Convolutional neural networks (CNNs), which are widely used and successful, were used in this study for feature extraction and image processing tasks. CNNs are particularly well-suited for image-related issues since they are excellent at extracting hierarchical patterns and correlations from visual input. Convolutional layers, pooling layers, and fully connected layers are the components of the network architecture that allow the model to effectively learn spatial hierarchies within the data (Jogin et al., 2018). These layers facilitate feature extraction and downsampling.

#### **3.6.2. Recurrent Neural Network (RNN):**

Recurrent neural networks (RNNs), which are intended to handle sequential data by maintaining memory of previous inputs, were utilized in this study to capture temporal connections between sequences (Schmidt et al., 2019). The RNN paradigm is particularly useful for jobs related to natural language processing, time series data, or any situation where input order matters. It effectively tackles the difficulties that come with working with data structures of this type.

#### **3.6.3. Graph Sample and Aggregated (GraphSAGE):**

Graph-structured data node-level classification challenges are the focus of graph neural networks such as GraphSAGE (Zhou et al., 2020). The approach shows successful generalization across bigger graphs by aggregating and sampling data

from nearby nodes. Applications like social network analysis and recommendation systems demonstrate how this strategy works best when nodes stand in for entities and edges for relationships.nearby nodes.

#### **3.6.4. Graph Convolutional Network (GCN):**

Neural network topologies called Graph Convolutional Networks (GCNs) expand the capabilities of traditional neural networks to handle graph-structured input (Liao et al., 2021). Node representations and their relationships are captured by GCNs through the use of graph convolution procedures. For tasks involving graph-based data, such chemical structures or citation networks, this paradigm performs very well.

#### **3.6.5. Hyper-parameter tuning for GCN:**

Hyper-parameter adjustment is used in this GCN version to improve the model's performance. The capacity of the GCN to recognize intricate patterns in data is maximized by methodically optimizing hyper-parameters, such as those controlling network architecture and training parameters. Increasing the model's accuracy and resilience in result prediction is dependent on this tactic.

#### **3.6.6. Graph Isomorphism Network (GIN):**

Graph Isomorphism Network (GIN) is useful for jobs where the topology of the graph is important since its goal is to find isomorphic structures within graphs. In order to accomplish this, GIN repeatedly aggregates data from nearby nodes, which gives rise to a more profound comprehension of the structural patterns found in the graph. This strategy has benefits for graph-based applications like chemistry and social network analysis.

#### **3.6.7. Graph Attention Network (GAT):**

By adding attention mechanisms, the Graph Attention Network (GAT) improves graph neural networks by enabling nodes to assess the significance of their neighbors during aggregation. The model is able to concentrate more on pertinent nodes thanks to this attention mechanism, especially in situations where some nodes are more important in determining the overall structure of the network. When it comes to jobs involving non-uniform graph designs, GAT performs remarkably well.

Chapter 4: Result and Discussion

The following accuracy are what I obtain after using the virus as the target variables:

	Model	Accuracy
0	CNN	0.926077
1	RNN	0.767015
2	GraphSAGE	0.968901
3	GCN	0.961509
4	GCN using hypertuning	0.966352
5	Graph Isomorphism Network	0.975784
6	Graph Attention Network	0.881213

Figure 4. 1: Accuracy of the models

The results indicate that the Graph Isomorphism Network (GIN) outperformed earlier models in malware detection with an impressive accuracy rate of 97.58%. Graph Convolutional Network (GCN) base and hyper-tuned variants achieved impressive accuracies of 96.15% and 96.64%, respectively, highlighting the importance of capturing graph structures for accurate malware detection. On the other hand, Graph Attention Network (GAT) obtained an accuracy of 88.12%, suggesting that using attention mechanisms for this task may have some limitations. Important insights may be gained by more examination of the particular features of the dataset that are causing the differences in model performance.

4.1. Precision, Recall, and F1 score:

Two metrics that are employed in the classification task are the f1 score and precision recall:

Precision:

The ratio of actual positive predictions to all expected positives is known as precision. The equation is:

**Precision = (True Positives)/ (True Positives + False Positives)**

Precision in malware detection measures how well the model can correctly identify some instances as malware out of all the instances that are predicted to be malware. Less false positives are indicated by a high precision, which is important when misidentifying benign files as malware is not what you want.

### **Recall (Sensitivity or True Positive Rate):**

The ratio of all actual positives to all true positive predictions is called recall. That formula is:

$$\text{Recall} = (\text{True Positives}) / (\text{True Positives} + \text{False Negatives})$$

Recall evaluates the model's ability to correctly identify every instance of actual malware in malware detection. Less false negatives are implied by high recall, which is important when it's crucial to detect any dangerous files.

### **F1 Score:**

Combining recall and precision into a single number is known as the F1 score. When it comes to binary classification problems, this harmonic mean of precision and recall is very helpful because it accounts for class imbalance. For the F1 score, the formula is:

$$\text{F1} = 2 * \{(\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})\}$$

When there is an unequal distribution between the two groups (malware and non-malware), a high F1 score indicates an equilibrium between precision and recall, which makes it very useful in these situations.

Model	Precision	Recall	F1-Score
Graph Convolutional Network	0.9853	0.9628	0.9739
Graph Convolutional Network (Hyper-parameter Tuning)	0.9871	0.9676	0.9772
Graph Sample and Aggregated	0.9808	0.9775	0.9791
Graph Isomorphism Network	0.9790	0.9887	0.9839
Graph Attention Network	0.9816	0.8569	0.9150
Convolutional Neural Network	0.9548	0.9457	0.9502
Recurrent Neural Network	0.8343	0.8583	0.8461

*Table 4. 1: Precision, Recall, and F1 score*

The efficacy of various malware detection models is displayed in the performance metrics table. With an F1-score of 98.39%, the Graph Isomorphism Network has exceptional precision (97.90%) and recall (98.87%). Similar to this, the Graph Convolutional Network has strong recall (96.28% and 96.76%) and precision (98.53% and 98.71%) in both its basic and hyper-parameter-tuned versions, yielding strong F1-scores of 97.39% and 97.72%, respectively. With an F1-score of 97.91%, the GraphSAGE model exhibits balanced precision (98.08%) and recall (97.75%). On the other hand, the Graph Attention Network (GAT) has a somewhat lower F1-score of 91.50% due to its lower recall (85.69%) and high precision (98.16%). In addition, the Convolutional Neural Network (CNN) outperforms the Recurrent Neural Network (RNN) in all metrics, achieving commendable F1-scores of 95.02% and 84.61%, respectively.



## Chapter 5: Conclusion and Future scope

We have discovered some fascinating findings from a variety of models in our investigation into Deep Learning for Malware Detection. The Graph Isomorphism Network (GIN) is the best performer with an accuracy of 97.58%, high precision (97.90%), recall (98.87%), and F1-score (98.39%). GIN's remarkable discriminatory strength is demonstrated by its capacity to understand complex connections seen in malware datasets.

By far the closest is the Graph Convolutional Network (GCN), which attains 96.15% and 96.64% percent accuracy in both normal and hyper-tuned versions. The models display noteworthy recall, precision, and F1-scores, highlighting the efficacy of graph-based techniques in comprehending the intricate relationships found in malware cases.

Graph Sample and Aggregated, or GraphSAGE, performs admirably as well, with a 96.89% accuracy rate and a precision rate of 98.08% that is perfectly matched with recall at 97.75%. Because it can efficiently combine information from nearby nodes in the network structure, it is an excellent categorization tool.

On the other hand, the Graph Attention Network (GAT) achieves an accuracy of 88.12%, offering a more complex viewpoint. Although its recall (85.69%) is lower than its precision (98.16%), this indicates that there may be difficulties in identifying specific malware-related patterns. This emphasizes how crucial it is to carefully examine the recall and precision metrics in order to carry out a thorough assessment of the model's performance.

Traditional models with accuracies of 92.61% and 76.70%, respectively, show competitive performance, such as the Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). CNN's strong performance can be attributed to its better spatial feature extraction capabilities compared to RNN, which include F1-score, recall, accuracy, and precision.

Ultimately, our results highlight the need for customized model architectures in malware detection applications. Among the many useful tools that demonstrate how graph-based techniques can be used to capture complex relationships seen in malware datasets is the Graph Isomorphism Network. As impressive as GCNs, GraphSAGE, and traditional models are in terms of performance, choosing the right model for a given task requires taking into account not only accuracy but also more subtle measures like precision and recall.



It is crucial for models to be both sophisticated and flexible given the ongoing evolution of malware threats. Important new information from our research will help this important field's future advances.

**Future Scope:**

Prospects for further developments in deep learning-based malware detection seem promising. Future studies ought to investigate ensemble approaches that combine the best model properties in order to increase accuracy. Attention processes may need to be adjusted in order to address the lower recall of Graph Attention Networks (GAT). Furthermore, for real-world implementation, it will be essential to optimize both computing efficiency and model interpretability. In the future, reliable defenses against changing cyber hazards will be shaped by cybersecurity systems that incorporate explainability, guarantee scalability, and adjust to new malware developments.

## Chapter 6: References

- Awan M.J., Masood O.A., Mohammed M.A., Yasin A., Zain A.M., Damaševičius R. and Abdulkareem K.H., (2021). *"Image-based malware classification using VGG19 network and spatial convolutional attention"*. Electronics, 10(19), p.2444.
- Azeez N.A., Odufuwa O.E., Misra S., Oluranti J. and Damaševičius R., (2021), February. *"Windows PE malware detection using ensemble learning"*. In Informatics (Vol. 8, No. 1, p. 10). MDPI.
- Bae S.I., Lee G.B. and Im E.G., (2020). *"Ransomware detection using machine learning algorithms"*. Concurrency and Computation: Practice and Experience, 32(18), p.e5422.
- Feng R., Chen S., Xie X., Meng G., Lin S.W. and Liu Y., (2020). *"A performance-sensitive malware detection system using deep learning on mobile devices"*. IEEE Transactions on Information Forensics and Security, 16, pp.1563-1578.
- Gibert D., Mateu C. and Planes J., (2020). *"The rise of machine learning for detection and classification of malware: Research developments, trends and challenges"*. Journal of Network and Computer Applications, 153, p.102526.
- He K. and Kim D.S., (2019, August). *"Malware detection with malware images using deep learning techniques"*. In 2019 18th IEEE International Conference on trust, security and privacy in computing and communications/13th IEEE international conference on big data science and engineering (TrustCom/BigDataSE) (pp. 95-102). IEEE.
- Hemalatha J., Roseline S.A., Geetha S., Kadry S. and Damaševičius R., (2021). *"An efficient densenet-based deep learning model for malware detection"*. Entropy, 23(3), p.344.
- Iadarola G., Martinelli F., Mercaldo F. and Santone A., (2021). *"Towards an interpretable deep learning model for mobile malware detection and family identification"*. Computers & Security, 105, p.102198.
- John T.S., Thomas T. and Emmanuel S., (2020, February). *"Graph convolutional networks for android malware detection with system call graphs"*. In 2020 Third ISEA Conference on Security and Privacy (ISEA-ISAP) (pp. 162-170). IEEE.

- Lee K., Lee S.Y. and Yim K., (2019). *"Machine learning based file entropy analysis for ransomware detection in backup systems"*. IEEE Access, 7, pp.110205-110215.
- Lee W.Y., Saxe J. and Harang R., (2019). *"SeqDroid: Obfuscated Android malware detection using stacked convolutional and recurrent neural networks"*. Deep learning applications for cyber security, pp.197-210.
- Ma Z., Ge H., Liu Y., Zhao M. and Ma J., (2019). *"A combination method for android malware detection based on control flow graphs and machine learning algorithms"*. IEEE access, 7, pp.21235-21245.
- Mahindru A. and Sangal A.L., (2021). *"MLDroid—framework for Android malware detection using machine learning techniques"*. Neural Computing and Applications, 33(10), pp.5183-5240.
- Mercaldo F. and Santone A., (2020). *"Deep learning for image-based mobile malware detection"*. Journal of Computer Virology and Hacking Techniques, 16(2), pp.157-171.
- Naeem H., Ullah F., Naeem M.R., Khalid S., Vasan D., Jabbar S. and Saeed S., (2020). *"Malware detection in industrial internet of things based on hybrid image visualization and deep learning model"*. Ad Hoc Networks, 105, p.102154.
- Pei X., Yu L. and Tian S., (2020). AMalNet: *"A deep learning framework based on graph convolutional networks for malware detection"*. Computers & Security, 93, p.101792.
- Pektaş A. and Acarman T., (2020). *"Deep learning for effective Android malware detection using API call graph embeddings"*. Soft Computing, 24, pp.1027-1043.
- Poudyal S., Dasgupta D., Akhtar Z. and Gupta K., (2019, October). *"A multi-level ransomware detection framework using natural language processing and machine learning"*. In 14th International Conference on Malicious and Unwanted Software, MALCON (No. October 2015).
- Qiu, J., Zhang J., Luo W., Pan L., Nepal S. and Xiang Y., (2020). *"A survey of Android malware detection with deep neural models"*. ACM Computing Surveys (CSUR), 53(6), pp.1-36.
- Ren Z., Wu H., Ning Q., Hussain I. and Chen B., (2020). *"End-to-end malware detection for android IoT devices using deep learning"*. Ad Hoc Networks, 101, p.102098.

- Roseline S.A., Geetha S., Kadry S. and Nam Y., (2020). *"Intelligent vision-based malware detection and classification using deep random forest paradigm"*. IEEE Access, 8, pp.206303-206324.
- Senanayake J., Kalutarage H. and Al-Kadri M.O., (2021). *"Android mobile malware detection using machine learning: A systematic review"*. Electronics, 10(13), p.1606.
- Shaukat K., Luo S., Chen S. and Liu D., (2020, October). *"Cyber threat detection using machine learning techniques: A performance evaluation perspective"*. In 2020 International Conference on Cyber Warfare and Security (ICCWS) (pp. 1-6). IEEE.
- Shaukat K., Luo S., Varadharajan V., Hameed I.A., Chen S., Liu D. and Li J., (2020). *"Performance comparison and current challenges of using machine learning techniques in cybersecurity"*. Energies, 13(10), p.2509.
- Singh J. and Singh J., (2021). *"A survey on machine learning-based malware detection in executable files. Journal of Systems Architecture"*, 112, p.101861.
- Ucci D., Aniello L. and Baldoni R., (2019). *"Survey of machine learning techniques for malware analysis"*. Computers & Security, 81, pp.123-147.
- Usman N., Usman S., Khan F., Jan M.A., Sajid A., Alazab M. and Watters P., (2021). *"Intelligent dynamic malware detection using machine learning in IP reputation for forensics data analytics"*. Future Generation Computer Systems, 118, pp.124-141.
- Senanayake, Venkatraman S., Alazab M. and Vinayakumar R., (2019). *"A hybrid deep learning image-based analysis for effective malware detection"*. J Inf Secur Appl 47 (2019) 377–389.
- Vinayakumar R., Alazab M., Soman K.P., Poornachandran P. and Venkatraman S., (2019). *"Robust intelligent malware detection using deep learning"*. IEEE Access, 7, pp.46717-46738.
- Wang S., Chen Z., Yan Q., Ji K., Peng L., Yang B. and Conti M., (2020). *"Deep and broad URL feature mining for android malware detection"*. Information Sciences, 513, pp.600-613.
- Wang W., Zhao M. and Wang J., (2019). *"Effective android malware detection with a hybrid model based on deep auto-encoder and convolutional neural network"*. Journal of Ambient Intelligence and Humanized Computing, 10, pp.3035-3043.
- Yuan B., Wang J., Liu D., Guo W., Wu P. and Bao X., (2020). *"Byte-level malware classification based on markov images and deep learning"*. Computers & Security, 92, p.101740.

- Yuxin D. and Siyi Z., (2019). *"Malware detection based on deep learning algorithm"*. Neural Computing and Applications, 31, pp.461-472.
- Zhong W. and Gu F., 2019. A multi-level deep learning system for malware detection. Expert Systems with Applications, 133, pp.151-162.
- Podder I., Fischl T. and Bub U., (2023, March). *"Artificial Intelligence Applications for MEMS-Based Sensors and Manufacturing Process Optimization"*. In Telecom (Vol. 4, No. 1, pp. 165-197). MDPI.
- Jogin M., Madhulika M.S., Divya G.D., Meghana R.K. and Apoorva S., (2018, May). *"Feature extraction using convolution neural networks (CNN) and deep learning. In 2018 3rd IEEE international conference on recent trends in electronics"*. Information & communication technology (RTEICT) (pp. 2319-2323). IEEE.
- Schmidt R.M., (2019). *Recurrent neural networks (RNNs): "A gentle introduction and overview"*. arXiv preprint arXiv:1912.05911.
- Zhou J., Cui G., Hu S., Zhang Z., Yang C., Liu Z., Wang L., Li C. and Sun M., (2020). *"Graph neural networks: A review of methods and applications"*. AI open, 1, pp.57-81.
- Liao W., Bak-Jensen B., Pillai J.R., Wang Y. and Wang Y., (2021). *"A review of graph neural networks and their applications in power systems"*. Journal of Modern Power Systems and Clean Energy, 10(2), pp.345-360.
- Wang Z., Chen J. and Chen H., (2021). *"EGAT: Edge-featured graph attention network. In Artificial Neural Networks and Machine Learning"*. ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part I 30 (pp. 253-264). Springer International Publishing.