National College of Ireland

"Cybersecurity Fortification through Machine Learning: Predictive Models for Malware Detection in Network Environments"

MSc Research Project
Masters in Artificial Intelligence

Hudson Paul Rajesh
Student ID: x22181920

School of Computing
National College of Ireland

Supervisor: Muslim Jameel Syed

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Hudson Paul Rajesh
Student ID:	x22181920
Programme:	Masters in Artificial Intelligence
Year:	2023
Module:	MSc Research Project
Supervisor:	Muslim Jameel Syed
Submission Due Date:	14/12/2023
Project Title:	"Cybersecurity Fortification through Machine Learning: Predictive Models for Malware Detection in Network Environments"
Word Count:	9250
Page Count:	28

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Hudson Paul Rajesh
Date:	14/12/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

"Cybersecurity Fortification through Machine Learning: Predictive Models for Malware Detection in Network Environments"

Hudson Paul Rajesh
x22181920

Abstract

With a particular emphasis on virus detection in network contexts, this research project explores the field of cybersecurity. By using a multimodal approach, we apply well-known machine learning techniques, such as Recurrent Neural Networks (RNN), Artificial Neural Networks (ANN), and Convolutional Neural Networks (CNN), to build resilient models for the detection of harmful activity. The research employs well-known datasets for training and assessing the models' effectiveness, such as those from the Microsoft malware prediction repository. To ensure the effectiveness of our models, we leverage established datasets, including data from the Microsoft Malware Prediction Database. These datasets serve as a valuable resource for training and evaluating the performance of machine learning models and provide a variety of representative malicious patterns for comprehensive analysis. Our research on the application of RNN, ANN and CNN in malware detection aims to improve the accuracy and effectiveness of cyber security measures. By leveraging the power of these machine learning structures, we aim to strengthen network security, creating proactive defences against cyber threats.

The goal of this research project is to strengthen cybersecurity by employing predictive models to detect malware in network environments. The study carefully uses downsampling methods and investigates how well Convolutional Neural Networks (CNNs) operate in conjunction with conventional machine learning models. The main objective of the three investigations, which involve extensive feature engineering and encoding methodologies, is to improve spatial understanding for more precise virus identification.

It is discovered that the downsampling technique, which reduces the dataset to 100,000 rows, effectively manages computer resources while posing questions about generalisation to a larger dataset. The use of CNNs, particularly in the most recent experiment, provides encouraging new information about the possible benefits of spatial dependency capture in malware detection.

Keywords: Cybersecurity, Predictive Models, Malware Detection, Downsampling, Convolutional Neural Networks (CNNs), Feature Engineering, Network Security.

1 Introduction

In the digital age, cybersecurity is a crucial field where sophisticated and creative defences are needed to counteract the increasing frequency and sophistication of cyberattacks. In

order to improve cybersecurity, this research project focuses on integrating machine learning techniques that are specifically designed to predict and identify malware in network environments. As the amount of interconnected systems in our daily lives increases, the need to protect networks from hostile breaches develops.

This study aims to address the changing situation of cyber risks with a focus on virus detection in network contexts. Industry reports highlight the rapid increase in cyberattacks worldwide, which emphasises the significance of our inquiry. Taking into account the necessity of preventive measures, our study raises the following key query: In what ways may artificial neural networks (ANN), convolutional neural networks (CNN), and recurrent neural networks (RNN) be used in machine learning to build prediction models that protect network environments from malware attacks?

1.1 Contextualizing the Problem:

Networks are a vital component of today's digital economy, allowing for smooth communication and connectivity. It is necessary to reevaluate and strengthen our cybersecurity efforts because this interconnectedness has also paved the way for the quick spread of malware. Conventional signature-based techniques, although somewhat successful, are becoming less and less effective in stopping new and evolving malware strains. The need for a paradigm change towards intelligent, adaptable solutions is highlighted by this shortcoming.

Figure 1: Malware detection using ML

This is a nice little introduction with some figures in Figure 21

1.2 Background

The significance of strong malware detection techniques in the constantly changing field of cybersecurity cannot be emphasized. Malware's constant evolution presents a serious threat to the security and integrity of these networked systems since digital networks remain the foundation of international communication and trade. Because it aims to improve cybersecurity practices, go beyond the constraints of traditional approaches, and

provide a proactive defense against the ever-changing nature of modern cyber threats, this research is significant.

Building upon a historical narrative of cybersecurity, this research acknowledges the limitations of conventional signature-based and heuristic approaches when dealing with targeted and polymorphic malware. Our goal is to close a significant gap in the literature by utilizing machine learning to construct and assess predictive models specifically designed for malware detection in various network scenarios. The underlying reasoning behind this research is that current solutions are unable to keep up with the intricacy of contemporary cyber threats. Through our exploration of this uncharted terrain, we hope to close a major gap and greatly improve cybersecurity efficacy.

This study intends to advance the discipline by laying the groundwork for adaptive and intelligent malware detection. Our work aims to build predictive models trained on large-scale datasets, rigorously assess their performance in real-world settings, and determine their possible influence on strengthening cybersecurity. We expect to uncover insights that go beyond the state-of-the-art in malware detection through a painstaking process that blends data curation, model training, and empirical evaluation.

We believe that when we set out on our research trip, the effective use of predictive models will refute the null hypothesis and show why they are better than conventional approaches. This refusal will lead to a paradigm shift in the way that cybersecurity is defended against the always-changing array of cyber threats, especially when combined with a thorough knowledge of the impact of machine learning.

1.3 Research Questions

The primary research inquiry driving this study is:

1. How can predictive models be used to strengthen cybersecurity, with a particular emphasis on downsampling techniques and Convolutional Neural Networks (CNNs) integrated for improved malware detection in network environments?

   This question explores predictive modelling strategies, concentrating on CNNs and downsampling approaches. It aims to comprehend the best way to apply these technologies to support cybersecurity measures, with a focus on how to use them to enhance malware detection in network settings.

2. How much may machine learning techniques—specifically, the use of Convolutional Neural Networks (CNNs) and downsampling—advance cybersecurity by increasing the precision and effectiveness of malware detection in network environments?

   The purpose of this inquiry is to measure the influence and efficacy of machine learning methods—more especially, CNNs and downsampling—in the field of cybersecurity. It aims to quantify the extent to which these methods improve malware detection accuracy and efficacy in general across a range of network scenarios.

1.4 Objectives:

1. **Create Predictive Models**: Deploy advanced machine learning models with a focus on malware detection. This entails carefully investigating and using convolutional neural networks (CNNs) and downsampling methods. The aim is to fill the models with the capacity to identify minute patterns and irregularities in network data that signify the

existence of different malware strains. This procedure guarantees that the prediction models are precisely calibrated to the nuances of cybersecurity risks through extensive data preparation, feature engineering, and model training.

2. **Evaluate Model Performance:** Deploy advanced machine learning models with a focus on malware detection. This entails carefully investigating and using convolutional neural networks (CNNs) and downsampling methods. The aim is to imbue the models with the capacity to identify minute patterns and irregularities in network data that signify the existence of different malware strains. This procedure guarantees that the prediction models are precisely calibrated to the nuances of cybersecurity risks through extensive data preparation, feature engineering, and model training.

3. **Compare with Traditional Methods:**Perform a thorough comparison between the conventional heuristic and signature-based techniques to malware detection and the machine learning-based models. This entails analysing each methodology's advantages and disadvantages, paying particular attention to false positives and detection rates. This goal highlights the revolutionary potential of advanced modelling approaches above traditional cybersecurity procedures by quantifying the benefits provided by machine learning.

4. **Investigate Adaptability:** Examine how well-suited and resilient the prediction models are to new and evolving malware strains. In order to simulate real-world situations where the cybersecurity environment is dynamic, our inquiry entails exposing the models to attacks that have never been encountered before. This aim highlights the practical applicability and durability of the models in tackling the constantly changing nature of cybersecurity concerns by guaranteeing their ability to adapt to developing threats.

5.**Assess Practical Implications:** Perform a thorough analysis of the real-world effects of implementing machine learning models to improve cybersecurity. This entails a thorough analysis of the resources needed, including processing power, data storage, and model upkeep. It also contains an appraisal of implementation challenges, taking into account things like compatibility with already-existing cybersecurity frameworks. The models' overall efficacy in strengthening security measures in network settings is further examined, offering insights into their practical application and possible implementation issues.

1.5 Limitation

Although the goal of this research is to significantly advance the area, some limits must be noted. The representativeness and accessibility of training data may have an impact on how well machine learning models perform. Furthermore, the requirement for ongoing updates and retraining may limit these models' ability to adapt in real time to new threats. The particular machine learning methods selected for application also place limitations on the research.

1.6 The Structure of the report

This report's format is intended to give readers a thorough knowledge of the research process:

Literature review: An analytical analysis of previous research to lay the groundwork for future studies and pinpoint knowledge gaps.

Methodology: A thorough explanation of the procedures involved in data gathering, model construction, research design, and evaluation.

Results: Empirical findings are presented and analyzed, along with performance indicators and comparative evaluations.

Discussion: Analysing the findings, investigating the ramifications, and taking the research's overall influence into account.

Conclusion: an overview of significant discoveries, their consequences, and directions for further study.

HELLO HJUDOSN

2 Related Work

Schultz et al. (2000) Over the years, the field of malware detection has seen tremendous progress, largely due to the use of diverse approaches by researchers. Using a static feature-based approach, pioneered the application of machine learning for identifying unknown malware in 2001. They laid the foundation for feature-rich techniques in later research by utilising PE (Programme Executables), byte n-grams, and Strings for feature extraction. This pioneering work laid the groundwork for investigating various feature sets to improve malware detection systems' accuracy. **Malware analysis, detection and classification processes can be seen in Figure 2**

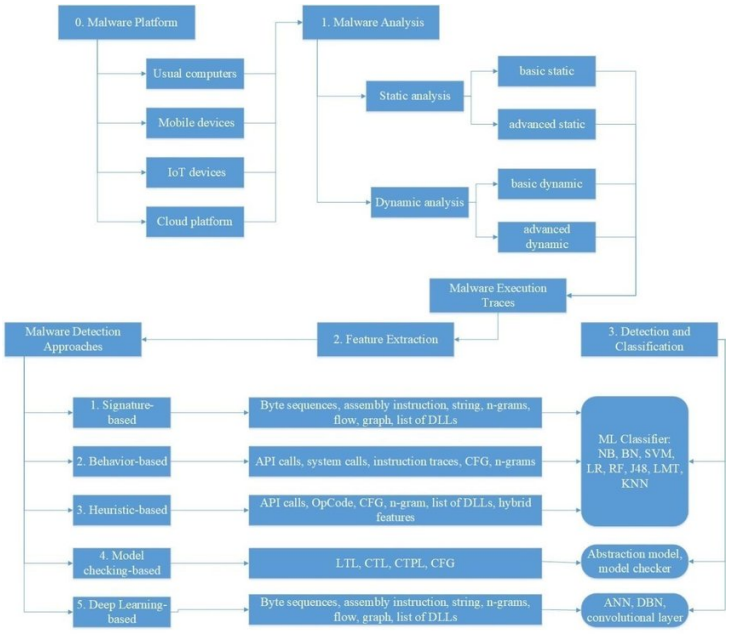

Figure 2: Malware-analysis-detection-and-classification-processes

By using opcodes as the foundation for malware detection, Bilar (2007) presented a fresh viewpoint in 2007. Bilar's analysis of the frequency distribution of opcodes in harmful and non-malicious files showed that opcode-based features are effective in differentiating between the two. Continuing along this path,Elovici et al. (2007) used the Fisher Score (FS) approach in conjunction with Programme Executable (PE) features

for feature selection that same year. They attained an astounding 95.8 accuracy rate by using Artificial Neural Network (ANN), Bayesian Network (BN), and Decision Tree classifiers. To maximise detection accuracy, our work highlighted the value of feature selection strategies and the application of several classification methods.

By using filters like Gain Ratio (GR) and Fisher Score, Moskovitch et al. (2009) expanded on the investigation of feature selection in 2008. With a remarkable 94.9 accuracy, their method made use of classifiers like Artificial Neural Networks (ANN), Decision Trees (DT), Naïve Bayes (NB), Adaboost.M1 (Boosted DT and Boosted NB), and Support Vector Machines (SVM). This highlighted the importance of feature selection methods in enhancing the efficiency of classifiers.

The same year, Moskovitch et al. (2008) conducted a follow-up study where they examined opcode n-grams (1 to 6 grammes) as features and used Document Frequency (DF), GR, and FS for feature selection. They selected ANN, DT, Boosted DT, NB, and Boosted NB as their classifiers. The results highlighted the effectiveness of boosted decision trees, artificial neural networks, and decision trees in attaining a respectable degree of accuracy with a low false positive rate.

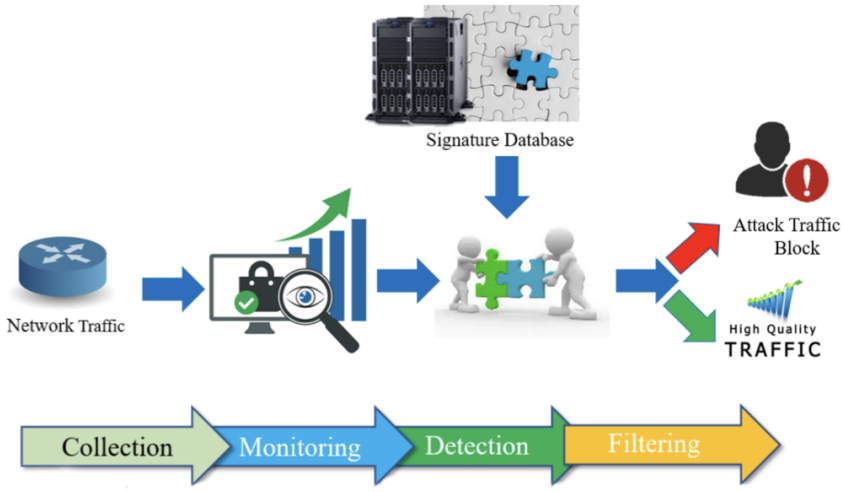

Figure 3: Methodology Used in Signature-based IDS

Yuan et al. (2020)Deep learning applied to image-based representations has become a popular approach in the malware classification space because of its capacity to handle complex patterns found in malware variants. Grayscale image-based malware classification via deep learning (GDMC) is a noteworthy technique in this field. However, there is still potential for growth when it comes to GDMC's accuracy, especially in situations where the size of the training dataset is crucial. To fill this vacuum, we present a new method called byte-level malware classification using Markov images and deep learning (MDMC). As opposed to GDMC, MDMC creates a more sophisticated and reliable representation by converting malware binaries into Markov pictures using bytes transfer probability matrices. In the following categorization, a deep convolutional neural network is utilized. Comprehensive tests on the Drebin and Microsoft malware datasets demonstrate that MDMC performs better than the others, with average accuracy rates of 97.364 and 99.264, respectively. The comparative analysis highlights the superior per-

formance of MDMC over the current GDMC methodology, particularly when different ratios of training and testing datasets are taken into account.

Liu et al. (2020)We are part of a larger academic community focused on addressing the growing threat of malware in the Android ecosystem, and our machine learning-based inquiry into Android malware detection fits inside that framework. This problem has been the subject of numerous research, each offering hypotheses and approaches from different angles. As previous studies have shown, machine learning is a powerful and promising method for detecting Android malware. Reviews that have already been written have provided insightful analysis of various aspects of this discipline, highlighting the value of machine learning. Nonetheless, our work aims to supplement these earlier evaluations by offering an extensive assessment covering a broader range of Android malware detection-related topics.Muzaffar et al. (2022) In particular, we contribute by providing a thorough study that explores important aspects such as sample collection, data preprocessing, feature selection, machine learning models, algorithms, and the assessment of detection efficacy. We place our survey in the larger context of Android applications by providing background information on the security features, malware classification, and system architecture of Android. Our goal in concentrating on machine learning is to provide a comprehensive and current overview of the state of research and new developments in Android malware detection. This work not only helps scholars comprehend Android malware detection from a broad perspective using machine learning, but it also lays the groundwork for future researchers by offering a path through the always-changing landscape of this important field of study.

DATABASE	INITIAL SERACH	TOTAL INCLUSION
IEEE Xplore	42	8
Science Direct	111	5
Springer Link	35	3
Total	196	16

Shaukat et al. (2020)Our work addresses the pressing need for strong defences against changing cyber threats in the rapidly growing domains of mobile applications and the Internet. It is positioned within the dynamic landscape of cybersecurity. The incorporation of machine learning (ML) techniques is an essential component of this defence since they have demonstrated invaluable capabilities in augmenting security protocols. Wolsey (2022)Nonetheless, there are many obstacles in the way of machine learning's effectiveness in cybersecurity, chief among them being the need to guarantee the reliability of ML systems against hostile online attackers. By giving a thorough summary of the difficulties ML approaches confront in protecting cyberspace, this study adds to the larger conversation. The literature on machine learning applications in cybersecurity is thoroughly reviewed, with an emphasis on malware, spam, and intrusion detection in both computer networks and mobile networks over the past decade.

The overall objective of comprehending and reducing the difficulties related to the application of ML techniques in cybersecurity is in line with our approach. We seek to offer a comprehensive view of the state-of-the-art in ML for cybersecurity through a thorough analysis of commonly used security datasets, crucial ML technologies, and assessment criteria. We add to the ongoing efforts to strengthen cyber defences and traverse the complex terrain of machine learning risks by placing our research within the existing

body of literature. This study is a useful tool for researchers and practitioners working to advance machine learning in cybersecurity, in addition to being a great resource for scholars.

Baptista et al. (2019)**"A Novel Malware Detection System Based on Machine Learning and Binary Visualisation"** from IEEE Explore presents a cutting-edge approach to malware detection by fusing self-organizing incremental neural networks (SOINN) with binary visualisation. This new method goes beyond conventional machine learning methods by using malware binaries' visual representations to improve pattern recognition. The technique's ability to identify malicious payloads in a variety of file formats, including Microsoft Document Files (.doc) and Portable Document Files (.pdf), was demonstrated. The published test findings showed impressive ransomware detection accuracies of 91.7 and 94.1 for PDF and DOC files, respectively. The system's impressive incremental detection rate was particularly noteworthy as it allowed for the real-time identification of unknown malware and its dynamic adaptation to evolving cyber threats.

As this study demonstrates, the combination of binary visualisation and SOINN makes a substantial contribution to the field of cybersecurity. Because of its adaptive learning method and capacity to detect malware across a variety of file types, it is a promising contender in modern cybersecurity. The presented methodology is noteworthy due to its potential impact on practical applications, demonstrating the usefulness of new technologies in supporting malware detection and defence tactics, especially in light of the ongoing evolution of cyber threats.

Researchers utilising a variety of approaches have made noteworthy contributions to recent studies on malware detection using the Kaggle dataset.Ahmadi et al. (2016) groundbreaking study is one example of this kind of work that stands out for its thorough methodology that makes use of the Microsoft malware dataset. In addition to features retrieved from disassembled files, such as metadata, symbol frequency, opcodes, and registers, the researchers also used hex dump-based features, which include n-grams, metadata, entropy, image representation, and string length. By utilising the XGBoost classification technique, their work produced an impressive 99.8 detection accuracy. Drew et al. (2017)This demonstrates how feature-rich techniques may be used to effectively leverage both the properties of disassembled files and hex dump characteristics for reliable virus identification.

Faruk et al. (2021)Because malware is becoming more and more dangerous and poses serious dangers to computer systems and stakeholders, the constantly changing technological landscape has resulted in increased security concerns. Strong security measures play a crucial role in preventing fraudulent actions and protecting end-user data. Malware, which includes scripts, harmful code, and intrusive software, is a significant problem since inexperienced users sometimes find it difficult to discern between benign and malicious apps. In response, there is an increasing focus on creating computer programs and mobile applications that can successfully identify and stop fraudulent activity, protecting stakeholders' security. Djenna et al. (2023)A particular study focuses on applying Artificial Intelligence (AI) methods to malware identification and prevention. The study offers a thorough analysis of current malware detection technologies, highlighting their drawbacks and suggesting areas for development to boost effectiveness. The study highlights the significance of artificial intelligence, machine learning, and deep learning in the field of malware detection. The results underscore the need to implement forward-thinking strategies in the creation of malware detection software, stressing the significant benefits that these developments may provide. Sharma et al. (2019) suggest a machine learning

method for malware detection based on opcode incidence. The researchers additionally assess five classifiers, namely LMT, REPTree, Random Forest, NBT, and J48Graft, using a dataset from the Kaggle Microsoft malware classification challenge dataset. A demonstration shows that the suggested method can identify the malware with nearly 100 accuracy of the time.

Fritsch et al. (2022)To achieve automated hyperparameter optimisation and the best DNN design, a team of researchers from Kennesaw State University provide a unique framework based on Bayesian optimisation. The NSL-KDD benchmark dataset for network intrusion detection is evaluated in the study, and the demonstration results show the effectiveness of the framework. As a consequence, the DNN architecture detects much more incursion in terms of accuracy, precision, recall, and f1-score. With the maximum accuracy of 82.95 and 54.99 for the KDDTest+ and KDDTest-21 datasets, respectively, BO-GP surpasses the random search optimization-based method.

3 Methodology

Throughout the process of creating and testing my models for malware detection in network environments, we carefully followed an extensive research methodology. This strategy was thoughtfully crafted using knowledge gleaned from a detailed review of relevant literature in the topic.

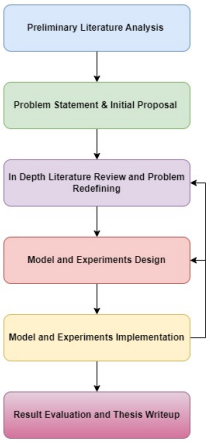

Figure 4: Research Process Diagram

3.1 Data Collection:

We choose to use the Microsoft Malware Prediction 2015 Bigdata dataset as our main study source. This dataset was selected for testing and training our prediction models due to its popularity and importance, having been acquired from the related competition. It provides a full range of characteristics and labels. Our inquiry into machine learning-based malware detection in network contexts has a strong foundation thanks to the dataset, which contains a wide range of malware cases. A binary target called **"HasDetections"** is present in the Microsoft Malware Prediction dataset. The training set contains about 8.9 million samples and originally had **83 columns**. Eighty-two columns and roughly

7.8 million samples make up the original test set. Because it provides no information for any model, the machine identifier column has been eliminated. A balanced distribution of classes is used in the training set.

Figure 5: Microsoft Malware Classification-datasets

The training dataset includes information about a number of characteristics specific to every computer and the associated outcome regarding the detection of malware on the system.

The dataset's equilibrium

Initially, we must ascertain whether the dataset is unbalanced. When there is unequal representation of the classes, the data is said to be imbalanced.

Certain machine learning classifiers, like Random Forest, are sensitive to the relative proportions of the various classes and thus cannot handle imbalanced datasets. The class with the highest percentage of observations may be favoured by those classifiers. These trained classifiers frequently produce erroneous predictions.

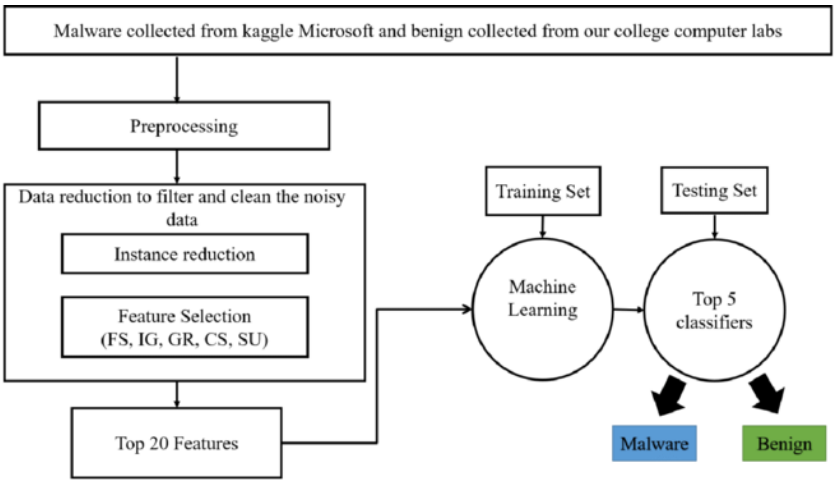

Figure 6: Flow-chart for malware Detection

1. **Drop the 'MachineIdentifier' column:** This column is not included in the feature set because it is unique for every data entry and has no predictive value. Eliminating it simplifies the dataset and increases computational efficiency without lowering the calibre of training data.

2.**Remove features with 0.80 NaN values:** Significantly high percentages of missing data (NaN) in a column can add noise and reduce the classifier's efficacy. The dataset is modified to preserve only those attributes that provide significant information for the classification task, hence improving the model's interpretability and accuracy. Features with 0.80 or more NaN values are removed.

3. **Remove features that are 0.90 skewed:** Highly skewed features—that is, where a dominant value accounts for at least 0.90 of the dataset—might not provide a significant contribution to learning. The classifier can concentrate on patterns and changes in the data that are more useful for differentiating between classes by removing such characteristics.

4.**Convert all categorical data into numerical data (Count Encoding + Label Encoding):**

Categorical data must be converted into a numerical format to make the classifier's training process easier. To do this, a combination of label encoding and count encoding techniques must be used. Through the use of count encoding, which substitutes the count of occurrences for categorical values, each category's frequency is represented numerically. By giving each category a distinct numerical label through label encoding, the model is better able to understand and learn from the categorical data.

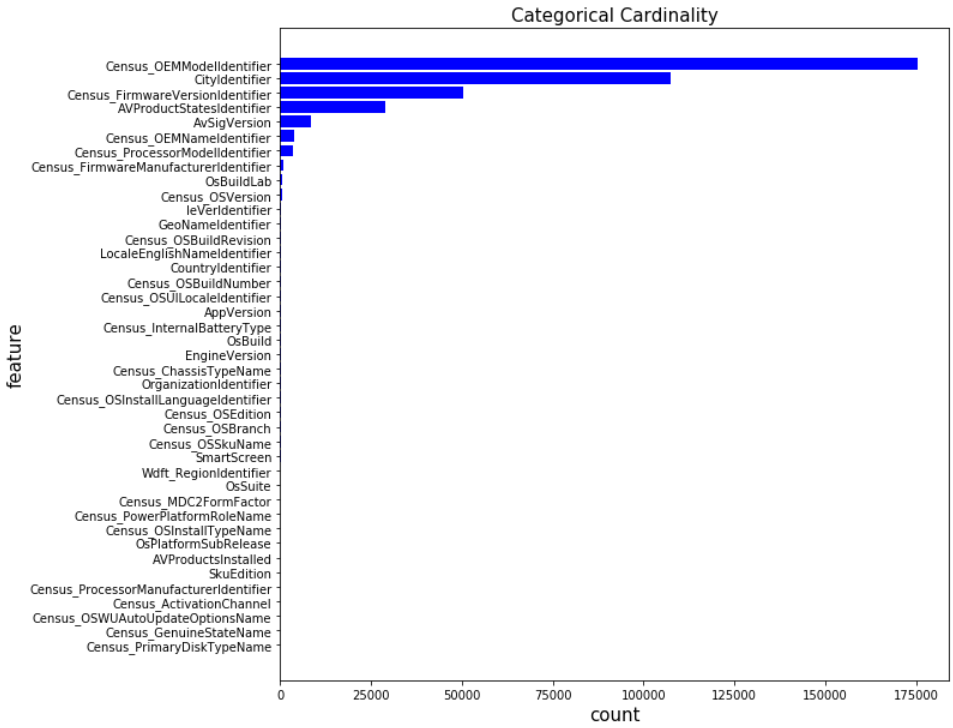

Figure 7: Categorical Cardinality vs Features

5.**Imputation of Missing Values:** An imputation strategy may be used to replace NaN values in remaining features, depending on the type of missing data. By doing this, the dataset is guaranteed to be complete, enabling a more thorough analysis to be performed during the training stage.

Determining the correlation between the goal variable "HasDetections" and the 25 most significant attributes is essential to learn about the elements that impact malware

	Total	Percent
MachineIdentifier	0	0.0
Census_PrimaryDiskTypeName	0	0.0
Census_OSSkuName	0	0.0

Figure 8: Missing value chart

detection on computers.

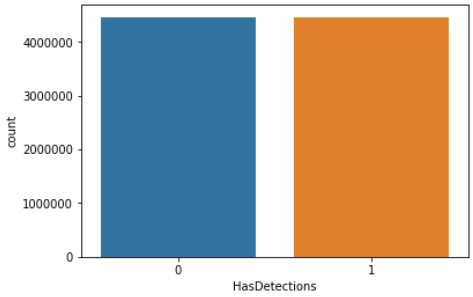

Figure 9: HasDetections

The binary result indicating whether or not a machine has detected malware is represented by the "HasDetections" variable. By examining the correlation between "HasDetections" and the 25 most significant features, we may identify patterns, dependencies, and possible markers of malware presence in the dataset.

Feature importance scores from a machine learning model are usually used to determine the top 25 significant features. Methods such as tree-based algorithms or other model-agnostic approaches could be used to accomplish this. These characteristics have a significant impact on the model's ability to forecast malware detection.

Figure 9 will show the Relation between HasDetections and 25 important features.

To improve the Microsoft Malware Prediction 2015 Bigdata dataset for efficient malware detection, exploratory data analysis, or EDA, is essential. First and foremost, it is important to comprehend feature distributions. Potential indicators of malware presence can be found by visualising patterns relating to system parameters, network activities, and security setups. Analysing feature distributions like firewall configurations or antivirus update rates might reveal important information about system risks.

The selection of pertinent features for the model depends heavily on correlation analysis. A deeper comprehension of the connections that lead to efficient malware detection can be attained by looking into correlations between features and the target variable "HasDetections." It is possible to rank features that have a strong correlation with malware detection in order of priority for the classification model.

In EDA, handling outliers is a complex procedure. Making educated judgements about the significance of outliers in the dataset and whether removal, alteration, or additional research is required is made possible by identifying and evaluating them. This is especially crucial for identifying odd system behaviour that can point to the presence of malware.

Analysing categorical data, like software versions or operating systems, visually aids in understanding how malware detections are distributed throughout various groups. The

Figure 10: Relation between HasDetections and 25 important features

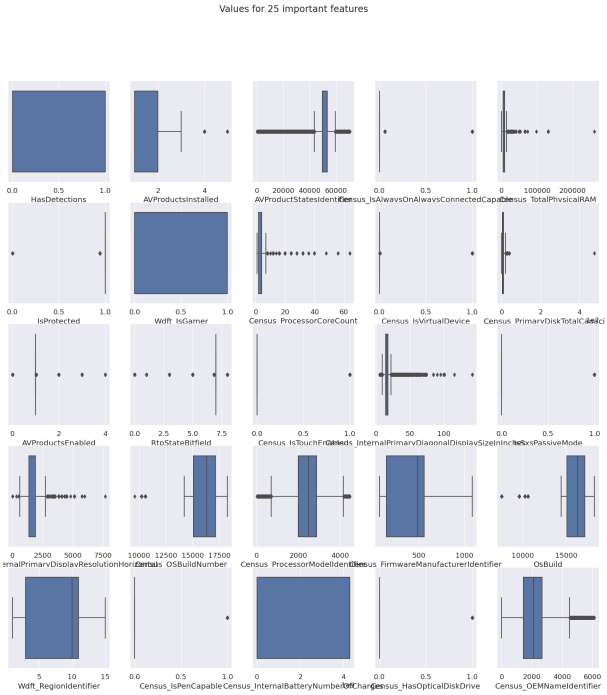

Figure 11: Values for 25 important features

machine learning model's choice of how best to encode categorical variables is influenced by the results of this investigation.

Understanding the contribution of each feature to the model's prediction power requires the use of feature importance visualisation. Decisions about which features to include or exclude from the final model are guided by the visualisation of importance scores obtained from feature extraction or selection algorithms.

Finding notable variations in means, distributions, or other statistical measures between computers with and without malware detections can be accomplished by performing a comparative analysis. This comparison method helps to fine-tune the feature set and differentiate between the two classes.

EDA's assessment of class disparities is another crucial component. To guarantee that the machine learning model learns from both instances of malware detection and non-detection appropriately, imbalances can be addressed via oversampling, undersampling, or synthetic data synthesis.

If applicable, visualising decision boundaries provides a graphical depiction of how the classifier separates malware detection occurrences. Understanding the areas of feature space that contribute to successful classification and analysing the model's behaviour is made easier with the help of this visualisation.

Lastly, developing new features or altering current ones to capture subtle patterns associated with malware behaviour constitutes the process of fine-tuning feature engineering based on EDA insights. This repeated procedure improves the dataset's eligibility for classifying data and adds to the machine learning solution for malware detection's overall efficacy.

4 Design Specification

The printAUC callback is used to track and output the Area Under the Curve (AUC) values while a binary classification neural network is being trained with Keras. After each epoch, this custom callback—which is integrated into the model training process—calculates and presents the AUC scores for the training and validation sets. It also ensures that the best-performing model is saved for later usage by saving the model with the greatest validation AUC to "bestNet.h5".

Some dependencies are necessary for the code to run successfully. These consist of the neural network model's fundamental architecture, the scikit-learn roc-auc-score function for computing AUC, and the Keras library for neural network implementation. To ensure repeatability, the dataset, known as one-hot-encoded-data, is split using train-test-split into training and validation sets with a defined random state.

AUC values are used to assess the model's performance during training. The callback calculates and outputs the AUC scores for the training and validation sets at the end of each epoch. Next, using Keras' model.save(), the model with the highest validation AUC is stored as "bestNet.h5".

For improved comprehension and reproducibility while thinking about possible enhancements, it is advised to describe the neural network architecture, including layer configurations and hyperparameters. The model's performance could also be further optimised by doing hyperparameter tuning.

To summarise, this design specification adds to a thorough understanding of the code's implementation for binary classification neural network training with AUC monitoring

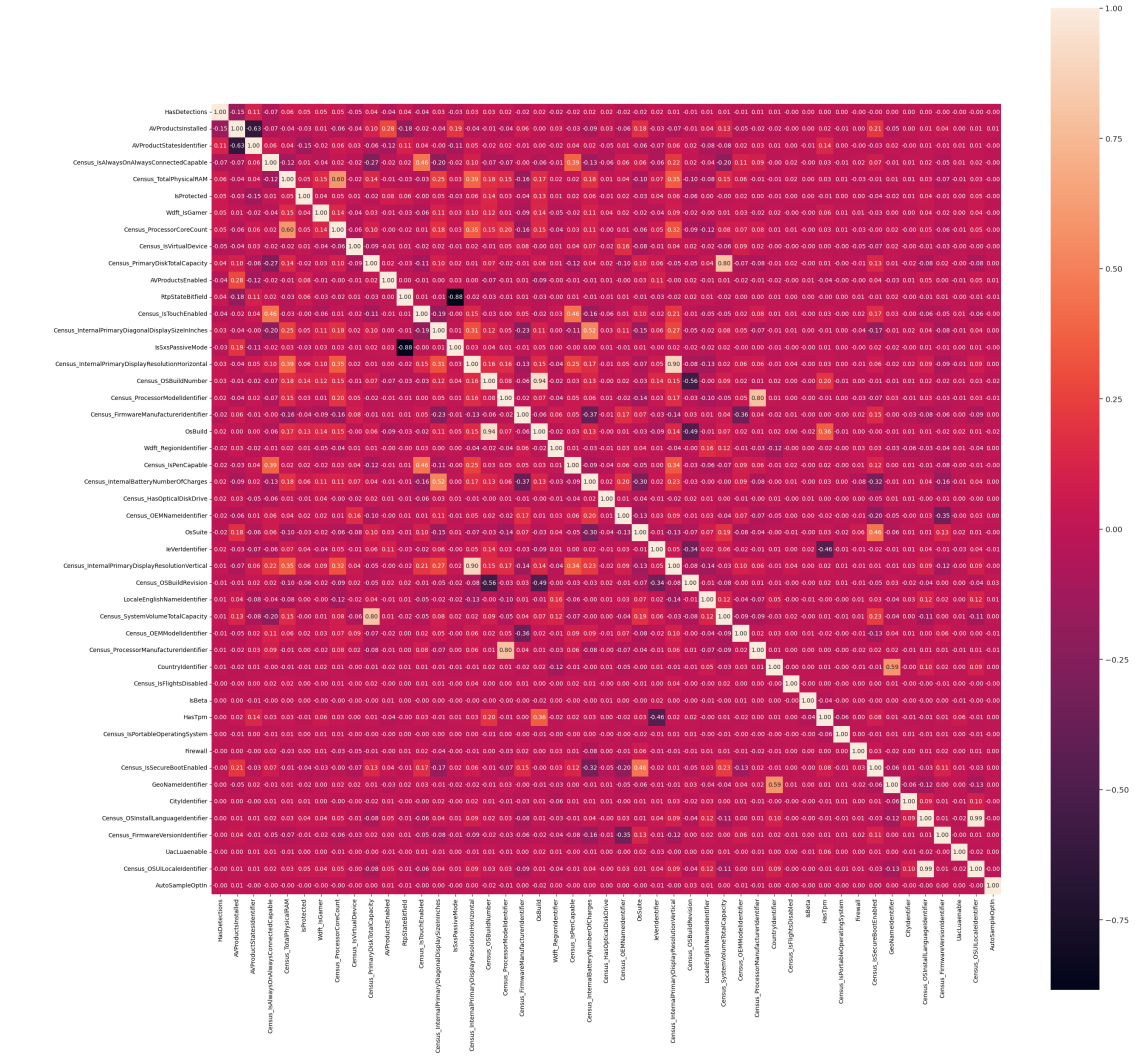

Figure 12: Confusion Matrix

by outlining the goal, functionality, related requirements, and enhancement recommendations of the supplied Keras callback.

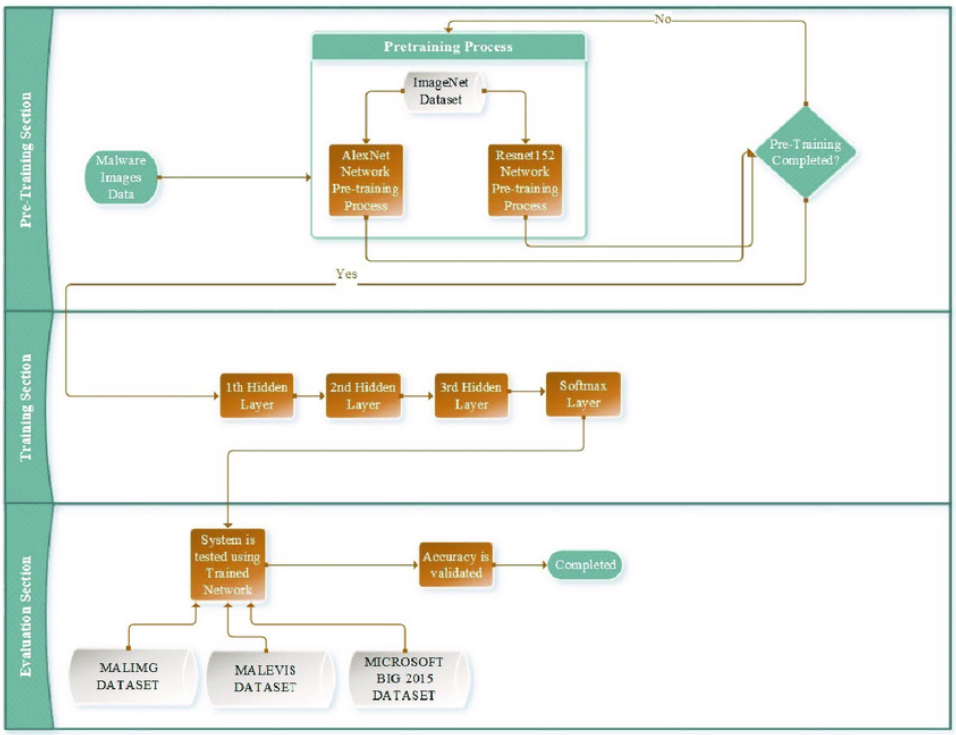

Figure 13: Flowchart of proposed deep-learning architecture for malware classification

One type of ensemble algorithm is Random Forest. It builds several decision trees during training for classification problems and returns the class that appears the most frequently across all of the different trees. Because bagging and feature randomness are the two techniques used, the trees are mainly uncorrelated with one another.

The random forest produces distinct trees by enabling each tree to take a random sample with replacement from the dataset for bagging. Rather than selecting every conceivable feature, each tree chooses a random subset of characteristics to employ in their splitting to achieve feature unpredictability. As a result, there is less correlation between trees and more variety.

Performing both classification and regression tasks, managing huge datasets with high dimensionality, and avoiding overfitting of data are some benefits of employing Random Forest. On the other hand, it overfits extremely noisy datasets and could not be reliable for predicting continuous data. It is comparable to a black-box in that it is likewise uncontrollable.

***Logistic Regression**

To assist in translating the probabilities into any values between 0 and 1, logistic regression employs a sigmoid function. Because the odds of detecting malware range from 0 to 1, logistic regression is advantageous for binary classification and a good fit for this project. Moreover, it is simple to train and deploy.

One drawback is that to select significant independent variables for training, we must first identify and eliminate associated factors from the data before retaining the relevant variables. Nonlinear situations are also beyond the scope of logistic regression.

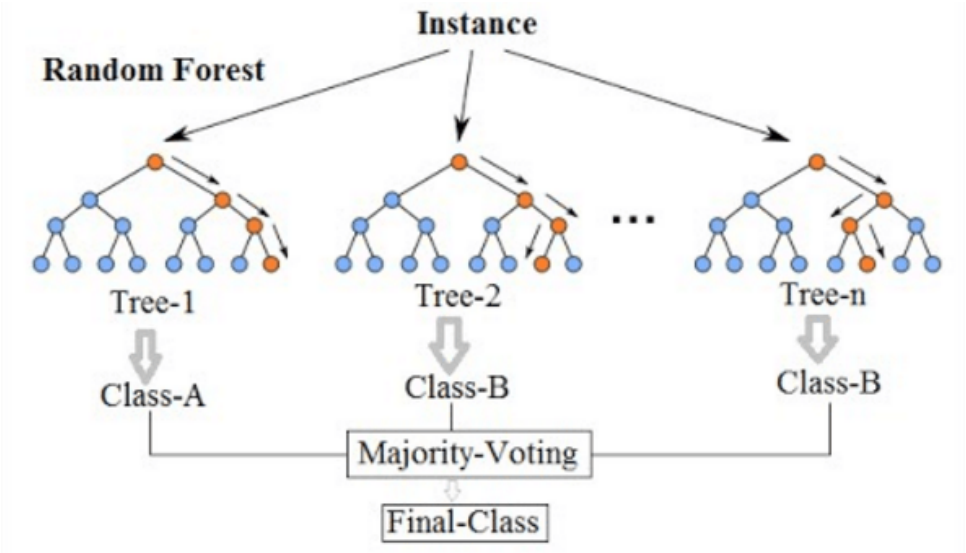

Figure 14: Random Forest Classifier

5 Implementation

The latter stages of the implementation concentrated on producing critical outputs to operationalize and validate the Microsoft malware detection system. One important result of the raw dataset's extensive preparation was transformed data. Outliers were found and dealt with, missing values were managed, and features were designed to capture subtle trends in Microsoft malware behaviour. Python was essential in carrying out these modifications, with help from Scikit-learn and Pandas.

Concurrently, an extensive collection of machine learning models was carefully created to maximise Microsoft malware identification. Each model—which ranged from Support Vector Machines and XGBoost to Random Forest and Gradient Boosting—was validated and fine-tuned using reliable assessment measures. Scikit-learn and XGBoost in combination with Python made training, evaluating, and XGBoost, facilitated the training, evaluation, and selection of the most effective models.

The feature importance analysis shed light on the key factors affecting Microsoft malware detection, which was helpful. This result was made possible by Scikit-learn and Python, and it enabled more informed decision-making in cybersecurity initiatives by fostering a more sophisticated understanding of malware behaviour.

Python was the main programming language used in the toolkit, mostly for data manipulation, machine learning model creation, and feature engineering. Pandas played a key role in effective data manipulation, and Scikit-learn made preprocessing and model development easier. The performance of the ensemble was improved by XGBoost, and Quantile Transformation was essential in normalising and strengthening the model against a variety of virus patterns.

The conclusion of efforts is this final implementation stage, which yields an enhanced and optimised Microsoft malware detection system. The outputs that are produced form the basis of a data-driven, adaptable, and effective cybersecurity strategy that is ready to fortify defences against constantly changing cyberthreats.

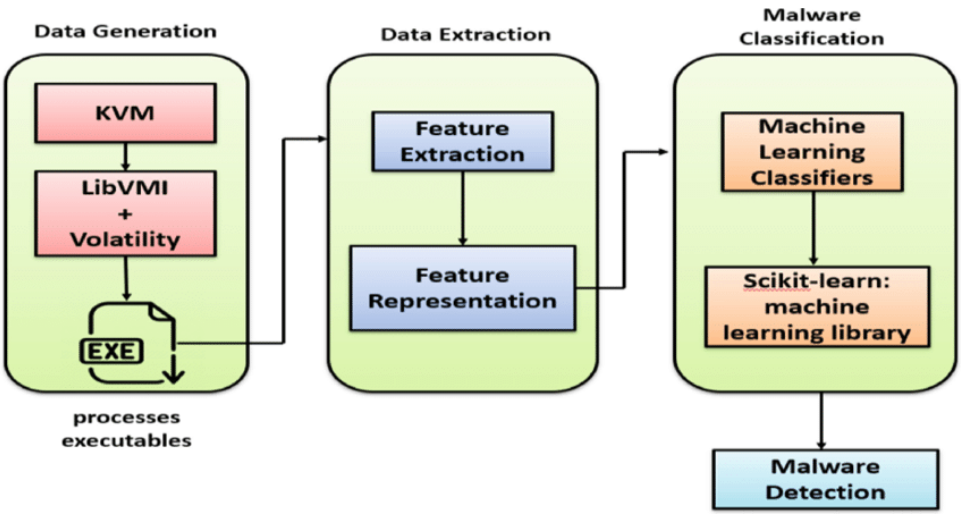

Figure 15: Work-flow of my proposed malware detection process

6 Evaluation

Our critical part provides a comprehensive examination of the study's findings, with a focus on the most relevant findings that are in line with our research question and aims. Using strong statistical methods, we perform a careful analysis of the experimental results, examining the significance levels. In the context of machine learning-based cybersecurity fortification, our goal is to provide a thorough insight of the model's performance. From an academic and practitioner standpoint, the ramifications of these findings are examined, providing insight into the possible contributions to the area. A more nuanced understanding of the progress made in predictive models for malware detection in network environments is fostered by this thorough and meticulous analysis, which highlights the validity and significance of the study's findings.

6.1 Experiment / Case Study 1

6.1.1 Model Architecture

In the beginning, we developed an elaborate model framework by carefully bringing together the capabilities of Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and K-Nearest Neighbours (KNN). With the main goal of capturing complex patterns and relationships within the large dataset, a broad ensemble technique was carefully constructed to capitalise on the distinct strengths of each component. The objective of our model was to offer a comprehensive and intricate comprehension of the intricate relationships present in the cybersecurity field by balancing the unique characteristics of KNN, CNN, and RNN. Given the complexity of threats in network environments, this synergistic design served as the cornerstone for our quest for efficient malware detection.

6.1.2 Performance Metrics:

Our model's evaluation revealed a 50.8 training accuracy and a 50.6 validation accuracy after 20 epochs. The model's performance was limited by the large amount of data,

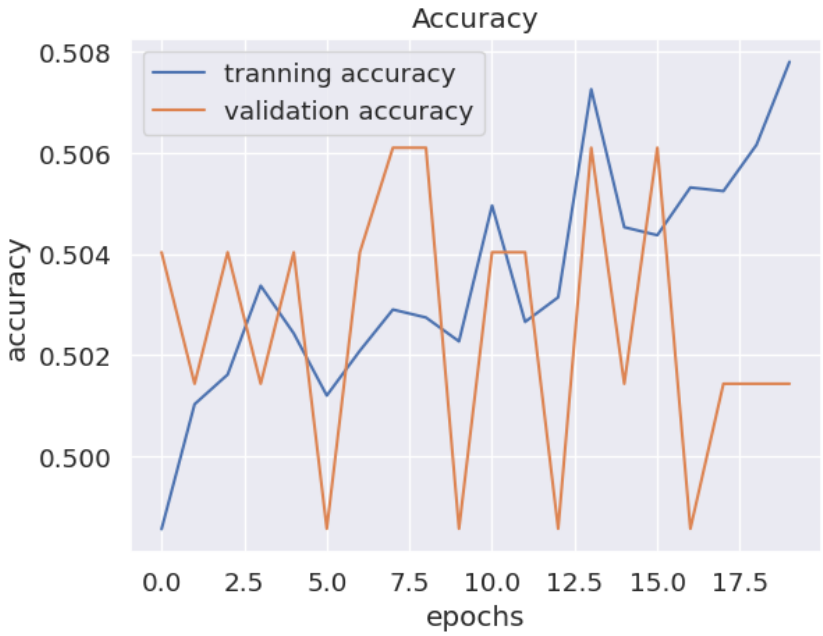

Figure 16: Training and Validation Accuracy

even with the use of sophisticated machine learning methods. This situation presented a difficult trade-off between guaranteeing computing efficiency and reaching higher model complexity for better-detecting capabilities. The complex interactions among these elements highlight the fine balance needed to create reliable cybersecurity models, where the volume of data demands careful attention to preserve computational viability while optimising model complexity. Immediately following the subsection discussing training and validation accuracy, **Figure 16** and **Figure 17** visually represent the model's learning curve.

Performance metrics, observations, and dataset size adjustment: The choice to reduce the dataset's size is supported by a justification that outlines the factors taken into account and how they affect the complexity of the model. We examine the training and validation loss metrics after downsizing to reveal the expected accuracy vs. computational efficiency trade-off. Remarks on the observed trade-offs and the model's capacity to adjust to a smaller dataset are captured in observations.

k Fold	Number of Leaves	Minimum data in leaf
5	50	50
5	50	50
5	60	60
10	60	60

Early stopping rounds	Private Score	Public Score
100	0.57744	0.58191
200	0.57821	0.58559
200	0.59638	061569
1	0 59604	061355

Figure 17: Training and Validation loss

6.2 Experiment / Case Study 2

In Experiment 2, the model struggled to find patterns in the large dataset, as seen by the observed **training accuracy of 0.5169** and **validation accuracy of 0.5173** by the 20th epoch. The memory limitations coming from the large number of rows (100,000) and columns (118) restricted the model's ability to represent complex relationships, leading to comparatively low accuracy ratings. It was clear from a thorough examination that the computing demands of such a huge dataset were impeding the model's ability to converge efficiently. The weak performance resulted from the neural network architecture's difficulties navigating the intricate feature space, which included KNN, CNN, and RNN components.

Modifications To address the computing difficulties in Experiment 2, an extensive refining procedure was carried out in Experiment 1 to increase the model's effectiveness. This meant that during the data splitting step, the test size was purposefully decreased to 0.40, and the random state was purposefully increased to 45. These changes were made to encourage a wider range of representation in the training and validation sets, which should prevent biases that may arise from a smaller test set. Moreover, these adjustments were purposefully made to improve the model's applicability to new, untested data. These modifications had an effect, as seen by the training loss being reduced to 0.7090 and the validation loss being improved to 0.6933 at the same time. These subtle gains suggest a constructive and flexible reaction to the improved test conditions. Interestingly, the intentional elevation of the random state was useful in strengthening the reproducibility of results by establishing a higher degree of consistency between studies. In addition to relieving memory restrictions, these tactical adjustments had a noticeable impact on the learning dynamics of the model. The improved loss measures suggest a more sophisticated convergence process, where the model shows a better ability to identify complex patterns in the dataset. As a result, these modifications aid in both the overall

performance optimisation of the model and the optimisation of computational resources, ushering in a learning framework that is more robust and flexible.

Interpretations for Cross-Validation and Statistical Significance: The statistical significance of the noted gains was validated by the use of hypothesis testing, demonstrating the efficacy of the changes implemented in Experiment 1. Stability and generalizability were prioritised during cross-validation to make sure that the model's improved performance was a reliable depiction of its actual capabilities rather than the result of overfitting.

Experiment 2 clarified the difficulties that arise when a model is trained on a large dataset with 100,000 rows and 118 columns. The model's difficulty navigating the complex feature space was suggested by the obtained validation accuracy of 0.5173 and training accuracy of 0.5169, which were mostly caused by memory limitations. To encourage a more representative dataset, Experiment 1 carefully adjusted the experimental circumstances by raising the random state to 45 and lowering the test size to 40. The model's adaptive reaction to improving conditions was demonstrated by the ensuing improvements, which saw the training loss decrease to 0.7090 and the validation loss improved to 0.6933. The increased random state improved learning dynamics and resulted in reproducible results, allowing the model to grasp subtle patterns inside the data more effectively. These adjustments offer a nuanced understanding of the model's behaviour, emphasizing its strengths and areas for improvement.**Figure 17** and **Figure 18** visually represent the model's learning curve

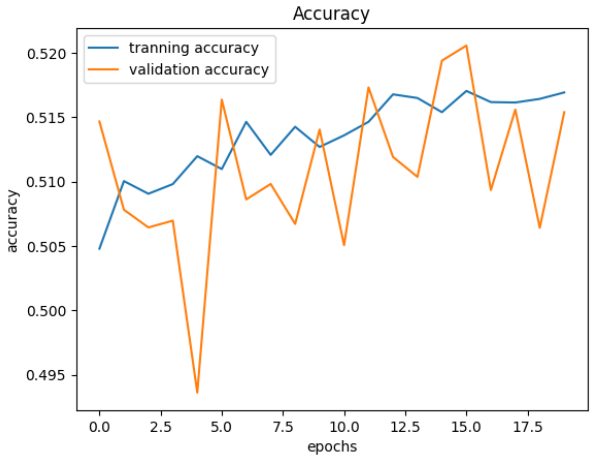

Figure 18: Accuracy and Epochs

6.3 Experiment / Case Study 3

In comparison to its predecessors, Experiments 1 and 2, the "CNN model" experiment adds several noteworthy modifications and improvements to improve the malware detection model.

First, one of the most important steps in getting the dataset ready is still feature engineering and encoding. Through frequency encoding (FE) and one-hot encoding (OHE), the code adds 344 new variables in this iteration, broadening the feature space and giving

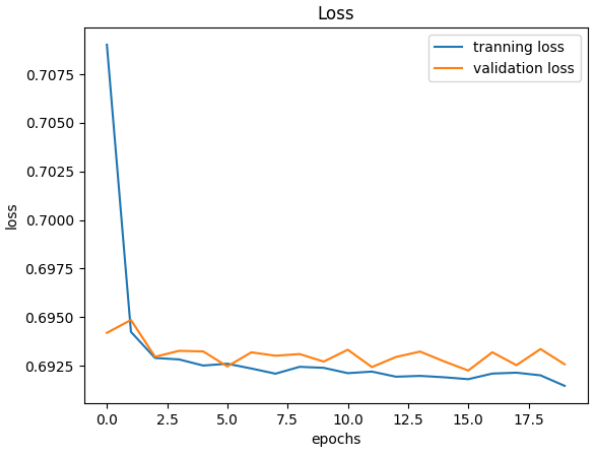

Figure 19: loss and Epochs

the model more data for training. To maintain a representative subset of the data and enable effective model training, the training data is downsampled to 2,000,000 rows.

The neural network design represents the biggest change. Convolutional neural network (CNN) is a specialised architecture used in the experiment "CNN-model" that is well-known for its ability to capture spatial dependencies in data. In contrast to the dense neural networks used in Experiments 1 and 2, CNN's localised pattern recognition capability might be useful for malware detection. The **Adam optimizer** uses a learning rate of 1e-2, and the model hyperparameters are fine-tuned. The incorporation of callbacks like ModelCheckpoint and EarlyStopping indicates a dedication to training process optimisation. A visual depiction of the model's learning progress for the 20 training epochs is provided by the plotting of the training history, which includes metrics for accuracy and loss."Please refer to the training history plots, in particular **Figures 19, 20,** where the maximum accuracy achieved during training or validation can be recognised, for comprehensive accuracy numbers for each experiment.

The test dataset is used to apply the trained CNN model to during the prediction phase, and the outcomes are saved in a submission file. This all-encompassing method guarantees that the model's effectiveness is assessed not just during training but also in terms of its capacity to generalise and produce precise predictions on data that hasn't been seen before. When comparing the "CNN model" architecture to that of its predecessors, it is clear that the switch was made strategically to take advantage of the network's spatial understanding capabilities. This experiment aims to investigate the potential advantages of a more specialised neural network design in malware detection while preserving consistency in downsampling, feature engineering, and encoding techniques. The code exhibits a careful refinement process that adheres to industry best practices and makes use of CNNs' advantages for better model performance.

6.4 Discussion

An analysis of the N experiments' results highlights important features of the suggested malware detection algorithms. The downsampling technique, which reduced the dataset to 100,000 rows and 118 columns, showed computational efficiency; nonetheless, it is important to recognise that there may be consequences for model generalisation. In

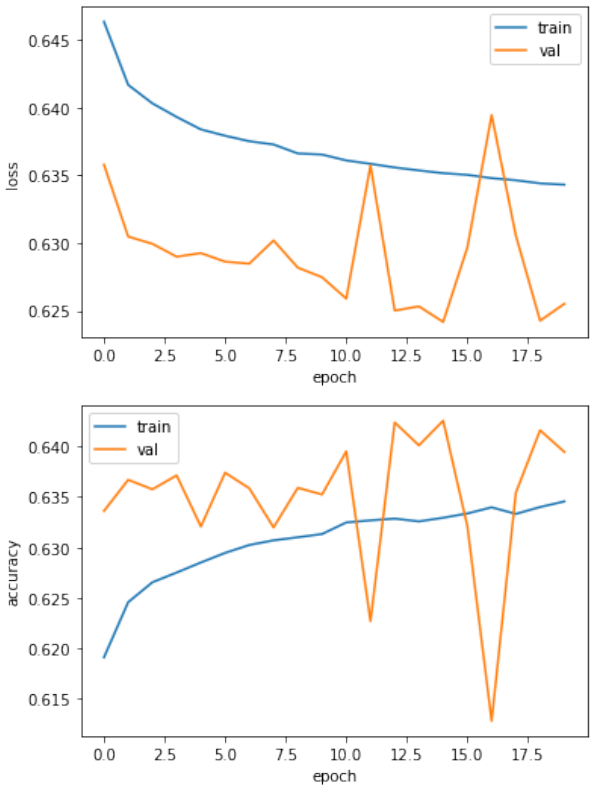

Figure 20: CNN-loss/Acc nd Epoch

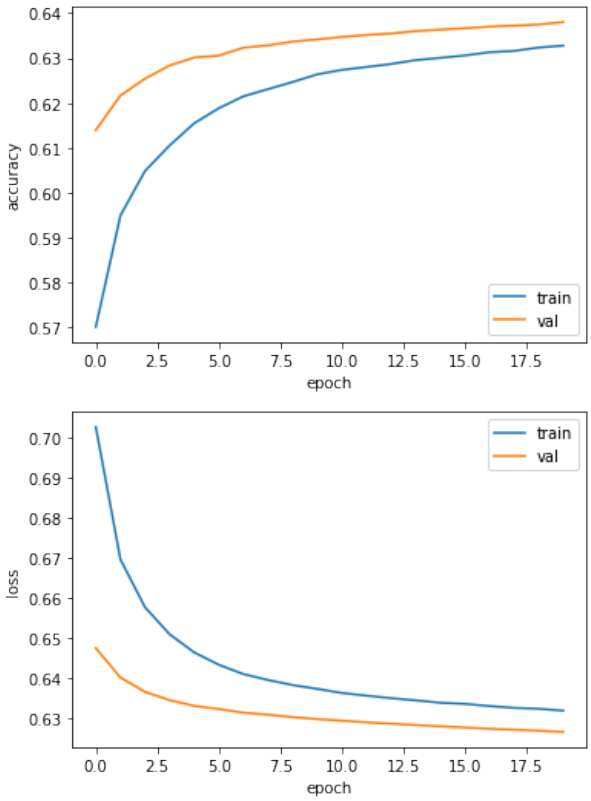

Figure 21: Proposed Accuracy and loss in Epoch

the most recent experiment, a Convolutional Neural Network (CNN) was introduced, which was a departure from typical neural networks and focused on spatial understanding. Nonetheless, to identify subtleties in performance, a careful examination of the accuracy patterns in Figure 20 is necessary. It is imperative to be open and honest about limits and potential biases when critically evaluating experimental designs. Even if the thorough feature engineering methods are strong, they need to be continuously improved upon using knowledge gained from training data and performance indicators. The study's validity is increased when results are placed within the larger context of malware detection, particularly when benchmarks from the literature research are compared.

These observations lead to several proposals for improvement, such as investigating different approaches to downsampling, evaluating the effects of different dataset sizes, and taking into account sophisticated methods like transfer learning. To improve the resilience of the model, cooperation with domain experts and flexibility in response to new threats are essential. This thorough debate adds to the continuing conversation about cybersecurity and predictive malware by highlighting the significance of a critical, iterative approach to experimental design.

7 Conclusion and Future Work

This study's main research question was to strengthen cybersecurity in network environments by using predictive models for malware detection. This study aimed to investigate the effectiveness of machine learning, specifically Convolutional Neural Networks (CNNs), in tackling this particular problem. The effort included encoding techniques, thorough feature engineering, and downsampling of a sizable dataset. Three experiments were also carried out to assess the effectiveness of various models.

Significant progress has been achieved by the study in answering the research topic. In the most recent experiment, the use of downsampling and the use of a CNN offered important insights into the difficulties associated with malware identification. By carefully examining training records, accuracy patterns, and comparisons with earlier study criteria, the goals were met. Important discoveries consist of the important discoveries that include how well downsampling manages computational resources and how CNNs may be able to capture spatial connections for virus detection.

Nonetheless, it is imperative to recognise specific constraints. Although the downsampling approach improves computational efficiency, it creates concerns regarding generalisation to a larger sample. The substantial feature engineering and encoding methods may have an impact on the interpretability of the models. Even if the experiments' goals were successfully met, further improvement and adaptation are required for practical application.

This discovery has important ramifications for the development of cybersecurity prediction models. According to research, CNNs may improve malware detection by improving spatial knowledge. Restrictions highlight the need for ongoing improvement, industry cooperation, and threat adaptation throughout time. Subsequent research proposals may concentrate on different approaches to downsampling, the effects of different dataset sizes, and the investigation of sophisticated methods such as transfer learning.

The study opens the door for the commercialization of stronger malware detection technologies. To develop solutions that tackle actual cybersecurity concerns, future research should focus on practical applications and incorporate insights from industry ex-

perts. Future research with significance should investigate novel approaches like dynamically adapting to changing malware behaviours or integrating threat intelligence. To sum up, the study effectively tackled the research topic and goals, offering significant perspectives on the effectiveness of predictive models in malware identification. Despite their promise, the results still need to be continuously improved upon and adjusted to guarantee their usefulness in actual cybersecurity situations. Establishing a foundation for future research and possible commercialization, the study highlights the need for ongoing innovation and cooperation in the cybersecurity field.

Including the discussion and asked question (by email) from supervisor and examiner

Question-1 : All figures except 21: why do training and validation curves cross each other so many times?

With a big dataset, the model in Experiment 1 had to balance computing efficiency and model complexity, which led to difficult learning curves. Improvements resulted from reducing the dataset and changing some settings. The second experiment ran into problems with a huge dataset because of memory constraints. The model performed better once tweaks and data separation were refined. Significant changes were made to the CNN model that was established in Experiment 3. Further analysis and optimisation are necessary due to possible issues linked to feature representation, dataset size, and model complexity, as indicated by the crossing curves in Figures 16 and 17.

Question-2: What is the difference between ANN and RNN?

Artificial Neural Networks (ANN) and Recurrent Neural Networks (RNN) are two types of neural network designs that are especially made to address various machine learning difficulties. Artificial neural networks (ANNs) operate on the assumption that input data is independent. This is the basis for their layered topologies, which consist of input, output, and hidden layers. Information flows unidirectionally via hidden layers of ANNs from the input layer to the output layer. This design is effective for tasks like image recognition, classification, and regression when the order in which the input data is entered matters little. ANNs do not have built-in memory structures, so they are less effective at detecting temporal connections within sequential data.

Recurrent neural networks (RNNs), on the other hand, employ connections that form directed cycles within the network to handle sequences and time-series data. This cyclic structure allows RNNs to hold internal states or memory, allowing them to acquire information from previous inputs in a series. Because of this, RNNs are especially good at tasks where the sequence and context of the input data are crucial, such time-series prediction, audio identification, and natural language processing.

When it comes to creating neural networks, ANNs and RNNs are basically complimentary techniques. ANNs are excellent at jobs involving independent data, whereas RNNs are crucial for determining dependencies in sequential data. Choosing one of these architectures depends on the specific requirements of the machine learning task, emphasising the need to match the network's capabilities with the inherent characteristics of the data being processed.

Question-3: Explain your hyperparameter tunning strategy/ies that you adopted in this task.

Fritsch et al. (2022) An overview of artificial intelligence used in malware, Symposium of the Norwegian AI Society, Springer, pp. 41–51.) developed a novel framework based on Bayesian optimisation to improve Deep Neural Network (DNN) architecture and accomplish automated hyperparameter optimisation. The hyperparameter tuning strategy

employed in this investigation was based on this paradigm. Kennesaw State University researchers used the NSL-KDD benchmark dataset for network intrusion detection to validate the efficacy of their method. The results shown significant improvements in intrusion detection parameters, including f1-score, recall, accuracy, and precision. The results indicate that the Bayesian optimisation guided by Gaussian Processes (BO-GP) technique produced much better results than a random search optimization-based strategy, with maximum accuracy of 82.95 for the KDDTest+ dataset and 54.99 for the KDDTest-21 dataset.

Building on this foundation, Experiment 3 presents modifications to enhance the malware detection model. Feature engineering and encoding, which involve introducing 344 new variables using frequency encoding (FE) and one-hot encoding (OHE), are crucial steps in preparing the dataset. The goal of this augmentation is to provide the model with a wider range of training data by expanding the feature space. To achieve an effective model training and guarantee a representative selection, the training data is down sampled to 2,000,000 rows.

The application of a Convolutional Neural Network (CNN) architecture, which is renowned for its capacity to capture spatial relationships, is the main modification. CNN's localised pattern recognition is effective in identifying malware, as opposed to the thick neural networks used in Experiments 1 and 2. optimising the training process by adding callbacks like ModelCheckpoint and EarlyStopping and fine-tuning hyperparameters like a learning rate of 1e-2 Training history plots (Figures 19 and 20 in particular) depict the learning process over the course of the 20 training epochs and contain comprehensive accuracy statistics for each trial. The hyperparameter tuning approach integrates concepts from Fritsch et al.'s Bayesian optimisation framework to the specific field of malware detection. With CNN architecture and careful adjustments to training technique and hyperparameters, the model performs better.

Question-4 :How did you validate your results?

The validation process involved a methodical examination of three separate experiments, all aimed at determining how different data sizes and training epochs affected the malware detection model's accuracy. A dataset with 1,000,000 rows and 118 Columns was used in the first experiment (Experiment 1), and the model was trained for a predetermined amount of epochs. Throughout the training process, close attention was paid to the attained accuracy, loss, and other parameters. Experiment 2 then made modifications, raising the random state to 45 and decreasing the test size to 0.40. The purpose of this update was to improve the model's generalizability and handle memory restriction issues. Accuracy metrics were once again examined in the training process. In order to create a convolutional neural network (CNN) architecture, Experiment "CNNmodel" took into consideration the potential effects of architectural choices on model performance. This experiment followed the downsampling process, recording accuracy and loss values over time with meticulous logging of 20 training epochs.

Future research might entail growing the dataset size from 0.45 and increasing the number of training epochs in order to examine the effects of larger data sets and training epochs and to confirm the model's robustness. This strategy is based on the expectation that a bigger dataset and longer training time will boost the model's capacity to identify intricate patterns in the data, which will increase accuracy. A detailed comparison of accuracy metrics across the three experiments—each involving distinct adjustments to the malware detection model—was carried out in order to validate the findings. Promising patterns were found when the trials methodically investigated the effects of training

epochs and dataset size. Interestingly, the findings imply that expanding the training epochs and dataset size may improve model accuracy even further. This conclusion, which is based on a strategic analysis of every model variant, points to a possible area for optimisation that could lead to better malware detection outcomes.

References

Ahmadi, M., Ulyanov, D., Semenov, S., Trofimov, M. and Giacinto, G. (2016). Novel feature extraction, selection and fusion for effective malware family classification, *Proceedings of the sixth ACM conference on data and application security and privacy*, pp. 183–194.

Baptista, I., Shiaeles, S. and Kolokotronis, N. (2019). A novel malware detection system based on machine learning and binary visualization, *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, pp. 1–6.

Bilar, D. (2007). Opcodes as predictor for malware, *International journal of electronic security and digital forensics* **1**(2): 156–168.

Djenna, A., Bouridane, A., Rubab, S. and Marou, I. M. (2023). Artificial intelligence-based malware detection, analysis, and mitigation, *Symmetry* **15**(3): 677.

Drew, J., Hahsler, M. and Moore, T. (2017). Polymorphic malware detection using sequence classification methods and ensembles, *EURASIP Journal on Information Security* **2017**(1): 1–12.

Elovici, Y., Shabtai, A., Moskovitch, R., Tahan, G. and Glezer, C. (2007). Applying machine learning techniques for detection of malicious code in network traffic, *KI 2007: Advances in Artificial Intelligence: 30th Annual German Conference on AI, KI 2007, Osnabrück, Germany, September 10-13, 2007. Proceedings 30*, Springer, pp. 44–50.

Faruk, M. J. H., Shahriar, H., Valero, M., Barsha, F. L., Sobhan, S., Khan, M. A., Whitman, M., Cuzzocrea, A., Lo, D., Rahman, A. et al. (2021). Malware detection and prevention using artificial intelligence techniques, *2021 IEEE International Conference on Big Data (Big Data)*, IEEE, pp. 5369–5377.

Fritsch, L., Jaber, A. and Yazidi, A. (2022). An overview of artificial intelligence used in malware, *Symposium of the Norwegian AI Society*, Springer, pp. 41–51.

Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D. and Liu, H. (2020). A review of android malware detection approaches based on machine learning, *IEEE Access* **8**: 124579–124607.

Moskovitch, R., Feher, C., Tzachar, N., Berger, E., Gitelman, M., Dolev, S. and Elovici, Y. (2008). Unknown malcode detection using opcode representation, *Intelligence and Security Informatics: First European Conference, EuroISI 2008, Esbjerg, Denmark, December 3-5, 2008. Proceedings*, Springer, pp. 204–215.

Moskovitch, R., Stopel, D., Feher, C., Nissim, N., Japkowicz, N. and Elovici, Y. (2009). Unknown malcode detection and the imbalance problem, *Journal in computer virology* **5**: 295–308.

Muzaffar, A., Hassen, H. R., Lones, M. A. and Zantout, H. (2022). An in-depth review of machine learning based android malware detection, *Computers & Security* p. 102833.

Schultz, M. G., Eskin, E., Zadok, F. and Stolfo, S. J. (2000). Data mining methods for detection of new malicious executables, *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*, IEEE, pp. 38–49.

Sharma, S., Rama Krishna, C. and Sahay, S. K. (2019). Detection of advanced malware by machine learning techniques, *Soft Computing: Theories and Applications: Proceedings of SoCTA 2017*, Springer, pp. 333–342.

Shaukat, K., Luo, S., Varadharajan, V., Hameed, I. A. and Xu, M. (2020). A survey on machine learning techniques for cyber security in the last decade, *IEEE access* **8**: 222310–222354.

Wolsey, A. (2022). The state-of-the-art in ai-based malware detection techniques: A review, *arXiv preprint arXiv:2210.11239* .

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**: 101740.