National College of Ireland

A Deep Learning Approach to Malicious Software Detection: Combining MLP and GRU

MSc Research Project
MSc cybersecurity

Ann Mohan
Student ID: X23175320

School of Computing
National College of Ireland

Supervisor: Jawad Salahuddin

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	ANN MOHAN
Student ID:	X23175320
Programme:	MASTERS IN CYBERSECURITY **Year:** 2024
Module:	MSC IN RESEARCH/ INTERNSHIP
Supervisor:	JAWAD SALAHUDDIN
Submission Due Date:	12-12-2024
Project Title:	A DEEP LEARNING APPROACH TO MALICIOUS SOFTWARE DETECTION: COMBINING MLP AND GRU
Word Count:	7909 **Page Count:** 22

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: ANN MOHAN

Date: 11-12-2024

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):	

A Deep Learning Approach to Malicious Software Detection: Combining MLP and GRU

Ann Mohan
X23175320

Abstract

The paper introduces the MLP-GRU model for detecting malware from two aspects, the MLP for network statics and the GRU for sequences analysis. The purpose is to increase the number of detections, minimize false positives, and increase workability for real-time and mass detection. Through the model, 99.4 % of classification was made, faster training particularly for big data, and better detection than existing models such as – Decision trees, SVM, and CNN. Two criteria, requirements and class imbalance, are met while building the model. For future work, the focus should be made on scalability, the training of the algorithm for optimization, as well as the application of transfer learning and automated feature engineering.

1 Introduction

1.1 Research Background

Malware is a short term of malicious software describing any kind of software designed with malicious intent toward the computer systems, networks, or users (Singh and Singh, 2020). Over time, malware has come in various forms, each with a different attack mechanism and impact. There are common types, which include viruses, worms and trojans. Every type of malware has different risks ranging from data theft, corrupting the system, violation of privacy, and loss of money (John Oluwafemi Ogun, 2024). Attackers have leveraged artificial intelligence, machine learning, and advanced encryption that makes it challenging to be detected and maximize the attack. In this scenario, effective malware detection became the prerequisite of cybersecurity (Ozkan-Okay *et al.*, 2024). Furthermore, malware attack causes direct and indirect losses such as; financial losses, leakage of information, privacy infringement, and embarrassment of national security. Traditional detection techniques rely heavily on signature detection methods where known malware is scanned through the comparison of code patterns or signatures held in a database (M. and Sethuraman, 2023). Heuristic and anomaly-based approaches tend to generalize poorly: while attempting not to miss subtle malware behaviors, they either overfit some malicious patterns-for instance, where this indicates a decrease in the effectiveness of these approaches across different malware varieties (Albakri *et al.*, 2023).

1.2 Problem Definition

The more advanced malware threats are, the more feasible, elastic, and efficient ML-based methods are for detecting and corresponding to the threats in a real-word environment (Hossain and Islam, 2024). Another major challenge thus arises in connection with the constant update of datasets which contain data on the new malware that are being created. The promising approach to cybersecurity that is particularly relevant for malware detection is deep learning algorithms because it can able to capture dynamic and intricate behaviors that commonly apply to malicious activities (Qureshi *et al.*, 2024). In malware detection, CNNs are applied to the binary representation of files where the files are treated as an image to identify structural patterns that might signify malware. However, in most scenarios, CNNs are not very suitable for tasks that should perform an analysis of data in sequence; in most malware detection this does not apply (Zhao et al., 2023). Malware can have specific characteristics and remarks for particular time or their sequence, for instance, system calls or network traffic. To address the sequence modeling need in malware detection, quite often, Recurrent Neural Networks are used. However, traditional RNNs suffer from the vanishing gradient problem in that the gradients shrink exponentially down a long sequence and are not capable of back-propagating sufficiently far for the model to be able to learn the dependencies along significant parts of the sequence. This is because of the limitation that results from the vanishing gradient problem which makes standard RNNs less effective. The MLP is a feedforward category of Neutral Network that consists of one or several layers of nodes which are completely connected and used in learning complex patterns in data. In malware detection, MLPs are employed to look for static patterns that may be inherent in the files and may indicate that malware executes (Hakim et al., 2024). On the other hand, even though MLPs dominate static feature detection in malware scenarios, they have a few boundaries. MLP, combined with models suited to time-dependent data, like Gated Recurrent Units (GRUs), may give a more comprehensive and robust defense against different kinds of malware threats because MLP deals with static analysis while other models capture the dynamic behaviours. Gated Recurrent Units (GRUs) is an extension of Recurrent Neural Networks created with the challenges in capturing long-range dependencies within sequential data (Som *et al.*, 2021).

Research Question 1. What is an effective approach to enhance malware detection systems using an MLP-GRU model that combines static pattern recognition with sequential data behavior analysis

The MLP-GRU model would be the most promising way of developing better mechanisms for malware detection: which means both static and dynamic analysis together would be more effective. The perceptions of the Multi Layer Perceptron part of the model are reliable in discerning the time invariants of the structural components or characteristics that are intrinsic to a pattern drawn from a malware file. Therefore, such static analysis appears to be a reasonable starting point for the identification of fundamental characteristics of malicious software. The component GRU solves a limitation of the staking techniques, to model sequential dependency and the kind of time-dependent behavior as evolvable malware activity.

From the mentioned perspective, MLP-GRU treats the two methodologies; MLP-GRU has dynamic sequence models like GRU put into it and in particular from those views. This compound scheme increases the efficiency of detection, flexibility, and redundancy to let the system detect and fight different types of malware in real-time. In the proposed paper a novel method of malware detection has been introduced which used MLP and GRU techniques together at one place which enhance the accuracy of the system as well as its stability. Within this context, the MLP can identify patterns that do not change inside a specific sample of malware, while GRU is employed in verifying states or behaviors through time. Based on the above incorporation through the integrated concatenation layer, the proposed model constructs a complete and extensible detection system that can effectively address the problem brought by polymorphic and dynamic malwares. The contributions include.

- A hybrid MLP-GRU model that combines static and sequential analysis for better malware detection.
- The malware detection systems should be robust and more adaptive to polymorphic and dynamic malware.
- Producing a comprehensive framework of classification with low false positives and high detection accuracy

Motivation of the Research:

The motivation for this research is due to the fact that malware is becoming increasingly complex, adaptive, and difficult for traditional detection methods to cater to. Using a hybrid model of MLP-GRU, this study aims for static and dynamic analysis into a robust, adaptive, yet accurate malware detection. These techniques are being used toward improving cybersecurity systems, lessening false positives, with an efficient counter to polymorphism and time-dependent malware attacks**.**

The remaining of the paper is as follows; Section 1 discusses the research context of malware detection in Cybersecurity. Section 2 discusses the related work accordingly regarding malware detection. Section 3 proposed the details of the used dataset, and methodology, such as implementing the integrated MLP-GRU model that is being discussed in this paper. Section 5 provides the implementation process and in Section 6 focuses on evaluation criteria applied. Section 7, consists of the overall conclusion of the study; the research limitations; and suggestion for further research.

2 Related Work

2.1 Machine Learning and Deep Learning Approaches in Malware Detection

Malware is seen as one of the biggest threats posed by usage of the internet by users today. Polymorphic malware is more versatile than the earlier forms of malware because it is

the third generation of the malware. It adapts this aspect at any one time camouflaging over its relied on key characteristics over a set period using standard signature-based approaches.

(Akhtar and Feng, 2022) compared different machine learning methods to identify malware and security threats. They discovered that the greatest approach with excellent results was one that incorporated a confusion matrix to differentiate between the false positive and false negative. They stressed the necessity for tracking and recognizing aggressive procedures on a computer network, in order to enhance its safety. The researchers applied correlation integrals of Machine Learning algorithm such as Naïve Bayes, SVM, J48 and Random Forest (RF and their method helps them to classify the malicious traffic in a more efficient manner. The classifiers applied in this study were DT, CNN, and SVM and the detection accuracy was 99%, 98.76% and 96.41% respectively. However, the study is not without limitations that include limited data sources available for collection, assessment of only static measures, which are different from the dynamic problems that can be experienced in real life. Such additional elements as dynamic analysis and others may contribute to the increased flexibility and the more stable work of the system. In the study conducted by Acharya et al., (Acharya *et al.*, 2021) outlined solution employing the Random Forest (RF) for the file classification and identification of the malicious files; signature matching for virus identification and the logistic regression for URL classification. The means of protection is overall based on the combination of measures used by the program but does not allow for versatile protection against different versions of malicious software. Greater model diversity and a dynamic analysis approach could help also system stability when threats appear as evolving over time.

Benign traffic is structured as symmetric, while malicious traffic is the opposite. With artificial intelligence methods it is possible to recognize and distinguish between different type of activities, namely, hostile ones and the legitimate ones. Alomari et al. (2023) suggest a stable malware detection model based on deep learning feature selection. Using two data sets they can distinguish between illicit and non-illicit activities happening in a network. This work seeks to apply dense layers and LSTMs in training deep learning models with the performance depending on the datasets employed. Nevertheless, the work lacks analysis of various malware types, as it considers the basics of these entities' identification. More specifically, the future research of the presented work can continue from where it left off, which involves considering different categories of malware and evaluating the effect that the feature selection had on the actual behavior of the malware samples. In another model for malware detection was proposed by Jeon and Moon (Jeon and Moon, 2020) where they use opcode sequences and a convolutional recurrent neural network. The model takes as input opcode sequences, and it extracts them from executables. The model performs well, passing 96% for malware; AUC of 0.99 and TPR of 95%. However, the given solution might not cover some aspects of malware features or might not have efficient solution for obfuscation problem. The same can also be said about assessment of the study's effectiveness using various real datasets. For future work, better adversarial examples against obfuscation should be explored and development of models should extend to a more diverse dataset.

Similarly, Jha et al., (Jha *et al.*, 2020) presents a new prognosis method called RNN for malware identification. The study examined an RNN with a directed graphical structure as well as the capacity to process temporal sequences. The study revealed that step size affects the classification results in a meaningful way. The Word2Vec feature vector, with Skip-gram architecture was seen to be even more efficient and stable having higher AUC and variance. However, the study has some limitations including excluding dynamic malware features and slow analysis for new threats. It is concluded that with dynamic analysis and real-life situations applied, the model would be more effective and stronger. The authors Oliveira and Sassi (Oliveira and Sassi, 2023) introduced a new approach to malware detectors using Direct Learning from API call sequences and behavioral graphs using Deep Graph Convolutional Neural Networks (DGCNNs). The performance of models based on DGCNN is slightly lower than LSTM but not dramatically, and the presence of a graph structure of API call sequences is critical for identifying malicious programs. In the future studies will estimate extended architectures and classify the malware into different sets according to the API call sequences and the corresponding behavioural graphs.

The study of (Lu et al., 2020) revealed that a real-world image classification utilizing both deep belief networks, the DBN with the incorporation of the GRU has been in the development to address the goal of Android malware. This method has been determined to have a higher accuracy as compared to other deep learning methods that was used before. The authors also mentioned that for detecting viruses in the Android platform, a deep network hybrid architecture was proposed was stimulated by the effectiveness of deep learning procedures in the tasks of feature representation learning. In another study conducted by Rimon and Haque (Rimon and Haque, 2022) by extracted and classified the feature and malware using the mixed deep learning approach. The hybrid method that that was used alongside the back propagation method with the addition of particle swarm optimization was accurate and required very little computation.

In the research done by (Chaganti, Ravi & Pham, 2022) introduces the DL malware classification and detection using Bi-GRU-CNN and the RNN for IoTs. The proposed approach was able to accurately detect malware at 100% while accurate classification of the malware families at 98%. The model was also very stable and platform independent and despite such complex input features such as byte sequences and CPU types it performed as an ideal. However, some essential classes of malware families were represented and not classified because of imbalanced databases. The results obtained from the model are contrasted with another model. There are many research works based on the anomaly malware detection in a similar way, the authors (Ullah, Mahmoud, 2022) suggested the anomaly detection which we proposed to be flow and flag features for the IoT networks, by employing the Feed-Forward neural network in the Limbic region. In which various types of network flow features are analyzed. The current research focus is directed to design and development detection of building emphasizing the importance of building design. system which is used to identify and detect the feasible and possible abhorrent activities for the IoT networks with the help of Feed-Forward. Neural work that is built on the executive flow and flag features. The additional search of this paper was to enhance the accuracy for the ML/DL. algorithms for the efficient

malware detection for SVM, NB, ANN, DT. The other proposal for the actual paper is Binary Classification and Multiclassification and with designed Random Forest Algorithm.

The approaches to mitigate the effects of Malware are also described in the research paper (Usman er al., 2021) as applied to the Malware IP problems and solution through CTI, Machine learning, Dynamic malware analysis and data forensics are also highlighted. He uses it to compare different machine learning approaches to get precision recall and a better f-measure. Decision tree techniques are better since analysis is inclusive and the percentage prediction is 93.5%. Real time analysis matches samples that cannot be recovered due to memory tampering, estimating professional variants deployed by malware. This brief includes the malware, dynamic, static, and family classification of analysis, evaluating the risk factors of the frequent malicious behavior as well as the observed behavior based on the paper. The findings of this study recommend implementing Kill Chain methodology for internal and external data after classification through machine learning malware detectors. In the study conducted by (Taheri et al.,2020) uses hamming distance to detect malware behavior. The results show that classifiers perform better than other algorithms, with a 90% higher accuracy rate than existing state-of-the-art solutions. The authors also propose a new system called Anastasia for identifying malicious samples through API features and system command. The study validates algorithms and proposes KNN-based solutions for malware detection.

(Yang, Zhao & Zeng, 2019) therefore developed a proposal for a model that embodied deep learning. algorithm, and in addition, the proposed approach also provided multidimensional features for the URLs. CNN Long This model was developed using Short-Term Memory Network algorithm. The model proposed comprised three which are the feature extraction, all the features were embedded and lastly the classification of the features. Then there is identification of relevant features, followed by the detection and further classification of the URLs. The results produced more precision against more standard computer learning methodologies and techniques. The false positive was also reduced in the model because of utilization of multidimensional features. In this paper, (Mustafa Majid et al., 2021) introduce what malware is and with it acknowledges the need for better methods of the effectiveness of malware detection using the AI and ML. They used the 3 algorithms of Convoluted Convolutional Neural Networks (CNN), Recurrent Neural Networks especially LSTM and auto encoders. Their research discusses Malware detection on Android OS, and they discuss how these 3 algorithms can be applied in order to detect malware. Their future scope asserted that auto encoders can have the most significant amount. promise as time goes by. The research performed by (Hadiprakoso, Buana and Pramadi, 2020) utilized the hybrid methodology in which they used static and dynamic features. They compared the deep neural network model with the models which were trained using machine learning such as Random Forest, SVM and Naïve bayes. They used two different models in deep learning such as DNN-S and DNN-D. Their research achieved higher accuracy compared to other machine learning models.

These reviewed papers present the advancement and issues of using machine learning and deep learning for the detection of malware. They demonstrate the importance of proper and creative approaches in the case of polymorphic malware. Several models and algorisms were

suggested including Random Forest, Naïve Bayes, Support Vector Machines, CNN, Recurrent Neural Networks and others are quite effective. From the findings of the study, techniques in the category of ML and DL may help enhance detection accuracy by about over 99 % precision. Yet there are still problems: small sets of samples for training, or non-stationary nature of malware activity. For these challenges, future studies should embrace multitype data, dynamic study, and improved adversaries while handling the mentioned problems. Thus, the study establishes that incorporation of dynamic and hybrid approaches, effective feature selection improvement, and sound adversarial techniques hold the key to improving malware detection.

3 Research Methodology

The approach for this research paper employs and combines deep learning, improving the efficiency and reliability of typical malware detection systems. The proposed architecture is a configurable one and it combines a Multilayer Perceptron (MLP) as a static domain processing module with Gated Recurrent Unit (GRU) as a sequential processing component. The process starts with data collection followed by preprocessing of data which includes data cleaning and normalization to ensure the quality and consistency of the input data. The MLP learns high-semantic level representations of static attributes, and the GRU captures temporal behaviors of sequential samples, providing substantial insights into the malware samples. These outputs are then combined through a concatenation layer, and then classify directly for the final prediction. The architecture of MLP-GRU is depicted in Figure.1.

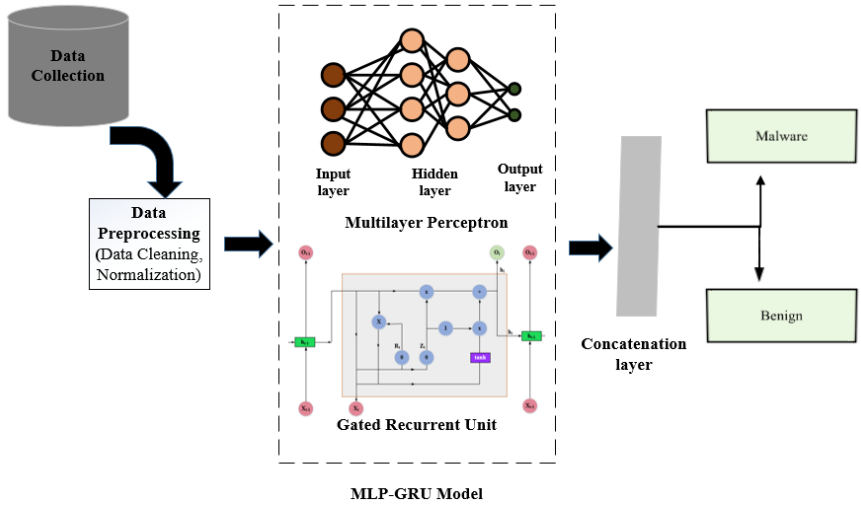

Figure 1: Architecture of MLP-GRU

3.1 Malware Detection-Data Collection

The malware dataset is obtained of Kaggle dataset. It is specifically aimed for malware classification and detection related researches. It comprises a set of encoded and marked malicious programs and their descriptors for recognition and assignment into specific

categories. These data points can be beneficial for machine learning tasks being related to cybersecurity and threats identification.

3.2 Data Pre-Processing

Data pre-processing is a process of selecting the data and preparing it for further analysis and data modelling. In other words, the main objectives are to eliminate the noisy data, meaning that it has to be refined and synchronized with other sources of data, its quality has to be improved and it has to be made ready for algorithms and models. Preprocessing, therefore, holds a significant contribution to the enhancements of the subsequently applied deep learning models. Record processing is defined as dealing with issues of missing characteristics, existence of duplicate records, and erroneous records. It can be filled using mean, median, mode imputation or some of the row with many missing value can be omitted. Such values are 'ROW 1, ROW 2' which are usually due to inefficiency in the structure or duplication of rows that reduces the efficiency of results or just provide unnecessary copies of data. Normalization is a process of making the values of variables pertaining to flow duration and packet rates of continuous data types have a value between 0 and one. This is very important in order not to end up with a situation where you have few features with large scales while other relatively small features are overshadowed in the model.

3.3 Feature extraction using MLP

An MLP stands for Multi-Layer Perceptron and this is a feed forward neural network which consists of many layers and contains all sort of connection that is from each neuron of the one layer to all neurons in the next layer. With respect to the data set input to the FNN, the construction of the MLP is designed for the learning of non-sequential static aspects of the dataset. The first type of flow features is static and does not depend on time or sequence (flow duration, total packets). These features are then put through a fully connected neural network which I will describe in the next section. In this case, the developed MLP is to explore the features which are extracted from the initial networks (such as flow duration and the count of packets & header length).

Input Layer: The input layer defines the characteristics of the dataset, the protocol being the total forward packets, flow duration and other traffic. The input layers take static features of a dataset.

Hidden Layers: In Hidden Layer, one or more layers are determined for implementing certain input features. In this case, every of these hidden layers consists of many neurons that calculate a weighted sum of their inputs along with an activation function that adds nonlinearity. About ReLU activation: ReLU activation removes linearity so that the network has the ability to learn complex patterns.

Output Layer: The output layer in the last of neural net and machine learning will provide the classification. If binary classification is applied for example malware and benign then the output will be between 0 and 1 by applying Sigmoid function. In the case of multi-class

classification, it will predict a probability distribution of multiple class by using the SoftMax activation. In the MLP, each layer analyzes the input raw network traffic data in a manner that increasingly defines and refines its input data into increasingly abstract functions.

3.4 GRU for Temporal Analysis

A type of Recurrent Neural Network, GRU is targeted at sequential or time-series data that normally exhibits temporal dependencies. It captures patterns and trends over time as malware behaviour does the same dynamically from bursts of activity or irregular packet patterns in a sequence of network traffic. The Update Gate and Reset Gate are the two major gates of the GRU. These gates control the information input and output, and so the information can be kept or reset according to what is read into the GRU.

Update Gate: This gate will define how much of the past information is to be retained in the present. If the information is relevant to a past state, it maintains it; otherwise, it doesn't consider it. Thus, it assists the GRUs in bearing some relevant information about the past network traffic, which might portray some anomaly.

Reset Gate: Some cells are connected to a reset gate that determines how much influence the past state should exert on the coming state calculation. It helps the model to forget unnecessary previous data, and therefore, pay attention to the current data. This assists in the development of rapid traffic spikes on a network which might be a sign of a possible break-out of malware.

The Candidate Hidden State: The candidate hidden state, which is the new state derived by combining the current input and the reset gate's influence on the previous hidden state to assist the GRU make rational prediction. It seems to be in some way a suggestion as to what the next state should be.

The main advantage of GRU is that it is good at capturing long sequences of inputs, making it efficient at detecting changes in malware behaviour over time.

3.5 Combination of MLP and GRU

To integrate the advantage of feature extraction and temporal behaviour capturing, the output features of MLPs and GRUs are optimized using a concatenation layer. This MLP, which looks at the static features of the input data returns a feature vector that represents high level non temporal characteristics. On the other hand, the temporal dependencies from the sequential nature of the data are captured by the GRU, which are important in capturing dynamic behaviours such as the activity of the malware over time. In the end the output of the MLP and GRU are concatenated into a single feature vector. In here, concatenating the outputs of the MLP and GRU models to form one large set of features. This combination enables the system to consider both static and temporal features while deciding. These combined features are useful in order to classify the software as malware or benign.

4 Design Specification

The tools that are presented in this context will be implemented ad hoc to achieve the purposes of the research and without needing particular hardware and software requisites. For the compulsory hardware requirements, there is an expectation of a minimum quad-core processor, including either AMD Ryzen 7 or Intel i7 processor, the RAM should be at least 16 GB and an optional Graphic Processing Unit such as NVIDIA RTX 3060 or higher to enhance the result of model training. On the software side, the operating system, on which the system is built, is Windows 10 (22H2 or any higher version), and Python version 3.9.13 as the programming language. Jupyter Notebook (v6.4.12) acts as the IDE in this work, as is a web application based on open-source web technologies that support numerous programming languages for adaptable and creative development. Other important frameworks and libraries applied are TensorFlow $\geq$ 2.4 for deep learning and Scikit-learn for machine learning, and Pandas, Numpy, Matplotlib, Seaborn for data handling, computation and data visualization respectively. Combining the hardware and software allows for the creation of a stable basis for the implementation, analysis and optimization of the implemented models of the experiment.

5 Implementation

The procedure of creating and using the malware detection system is as follows: environment, where Anaconda was downloaded and configured, the FN Control Center was launched together with Jupyter Notebook. The data preprocessing includes the downloading a labeled Malware Dataset, necessarily implemented libraries like pandas, NumPy, TensorFlow, scikit-learn, matplotlib, seaborn and other tools. The first step of the analysis is the data collection where one ensures that the collected data has training data.

The second process is data preprocessing where data cleaning is done by eliminating rows with more than one instance or none at all, using Label Encoding to normalize object label features, handling missing values and eliminating duplicate entries. This step also involves using bar charts for category distribution and using encoding labels for frequency counts. This operation is then followed by feature engineering, which involves using heat maps for correlation analysis to eliminate such features, feature scaling to normalize the ranges of such features, and merging of the minority classes together, and SMOTE to oversample such classes. The process of feature scaling and normalization makes analysis consistent and ensures that all features are of similar magnitude thus making the model perform well.

Model architecture consists of two main components: for this, I designed an MLP for static analysis and a GRU for sequential analysis. The MLP input layer has neurons equal to the feature count; hidden layers are fully connected efficiency with ReLU activation; the last stage is the output layer in which MLP maps intermediate feature representation. The GRU component takes sequential input and passes it through 64 GRU units and has the added bonus of a dropout layer to stop overfitting. The final layer joining MLP and the output vector of GRU. The final layer oxide used is SoftMax for multi-class classifications whereas binary use oxide uses sigmoid. When doing the final model training it then proceeded with 10 fold cross validation process, and then split the data into training, validation and test data. In the 10th epoch, training is carried out with a test loss of 0.0327 and an accuracy of 99.40%, which supports a prediction dominance model. The actual and detailed results of the classification

report provide such detail outcomes including the precision, recall and F1 Score of each of the class as well as support with the macro average and the weighted average. Therefore, I utilize the ROC-AUC curve for plotting the performance of the model in its classification between two classes and F1-score in a form of a bar chart. Finally, prior to model selection, a Cumulative Gain Chart is drawn again to verify the model's proficiency in identifying relevant samples better than random selection providing a third instance of confidence.

Altogether, the developed MLP-GRU model is effective in malware detection due to its modular architecture, extraction of static and sequential feature vector, feature engineering, data preprocessing, balancing approach, and the proposed model evaluation plan discussed in this paper.

6 Evaluation

Thus, this research confirms that employing the suggested MLP-GRU model as the classification method is more effective in improving the classification rates and useful in addressing imbalanced data sets than conventional models. Basic criterion, namely accuracy, precision as well as recall and F1-score where used to verify the validity of the proposed model, as well as its comparison with traditional approaches. Such finely tuned combinations as MLP-GRU are depicted from these results to be efficient for such complex operations as classification procedures. In addition, generalization capability of the model when coupled with the fact that the class distribution is skewed towards one class in real life makes the model very useful. Those crucial factors including scalability, the computational performance, and model interpretability are a basis for further research directions for making effective predictive machine learning models. The comparison of the proposed method with more traditional methods is discussed in order to pinpoint the potential of the MLP-GRU model for improved classification of the dataset and the remedy of the problem of the dataset’s imbalance.

6.1 Class Distribution

Figure.2 represents the distribution of a class for the data set. Here the x-axis denotes five classes of which the class label are represented as 0, 1, 2, 3 and 4 respectively. The y-axis on the content is the frequency and this indicates the number of items that were in a particular class. In this case, class 1 evidently dominates the majority of the distribution shares with a hyper-frequent frequency value more than 800,000. Another evidence drawn from the plot is that the other classes are way much fewer.” Class 0 is detected as the second most frequently, while classes 2, 3, and 4, however, frequent approximately as Classes 2, 3, and 4. This again is a good pointer that it goes with a high majority from Class 1 and from the other classes a few. This can become an issue in the task of machine learning; to handle class imbalance typically techniques such as oversampling, undersampling, or the use of weighted loss functions need to be employed to balance performances of all classes in the model.

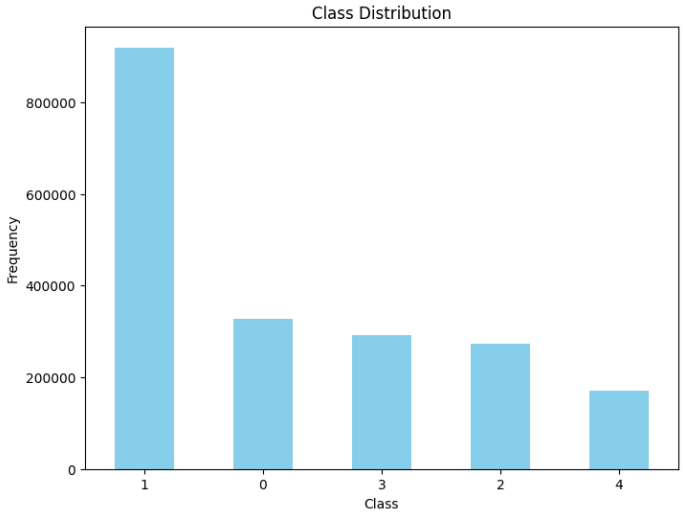

Figure 2: Class Distribution

The Figure 3, titled "Class Distribution After Down sampling" presents the class distribution of the five classes, Category 0, Category 1, Category 2, Category 3, and Category 4, following the application of down sampling. Each category is represented by a certain colour: blue for Category 0, orange for Category 1, green for Category 2, red for Category 3, and purple for Category 4. About 250,000 counts for each type is shown on the y-axis. This act of down sampling prevented a bias in any model concerning specific categories. Such equilibration reached makes it able to make equally good inferences on all such categories; thus, its predictions would be more precise and fair-minded.

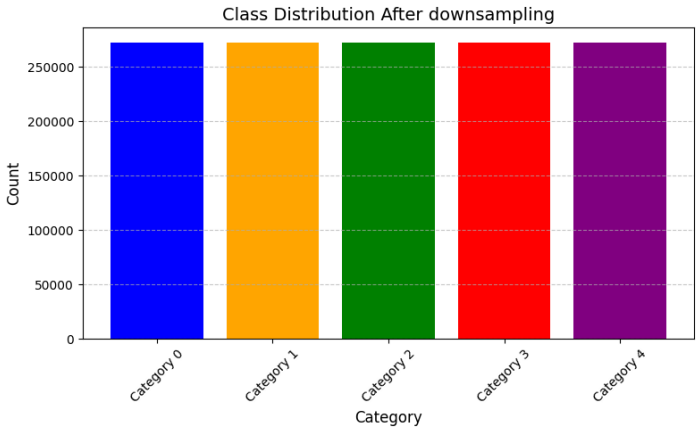

Figure 3: Class Distribution After Downsampling

The Figure 4, Model Accuracy shows the performance of a model in percentage for 10 epochs or rounds of training. The x-axis represents the number of epochs, or training iterations, while the y-axis measures the model's accuracy, indicating its performance. The graph features two distinct lines: Training accuracy is plotted in blue line whereas the validation accuracy is plotted in orange line. The training accuracy initially is about 0.965 and gradually increases throughout to touch 0.990 plus by the 10th epoch. On the other hand, validation accuracy starts form as high as 0.990 and gradually increases to just below 0.995 at last epoch. This trend

reveals that an increase in the training accuracy coupled with a concomitant increase in the validation accuracy indicates a good model learning and validation. Surprisingly, there is still a little difference between the validation accuracy and the training accuracy over the epochs and this means that the model does not fully memorize the data and is unused on unseen data.

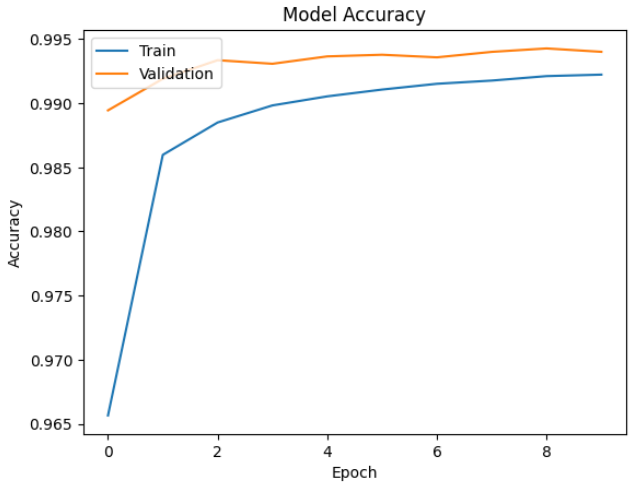

Figure 4: Model Accuracy

The Figure 5, graph titled "Model Loss" shows the training and validation loss of a model in 10 epochs of deep learning. The horizontal axis refers to epochs while the vertical axis illustrates loss and is between 0.00 and 0.14. The first y-axis blue line corresponds to the training loss starting from 0.13 which quickly drops down to 0.04 in the second epoch. The orange line of graph indicates the validation loss commences from 0.04 and does not fluctuate much just has a slight drop which is always below the training loss. This shows that the model is capable of learning the data well and at the same time, is capable of performing well on new data without getting mislead by noise.

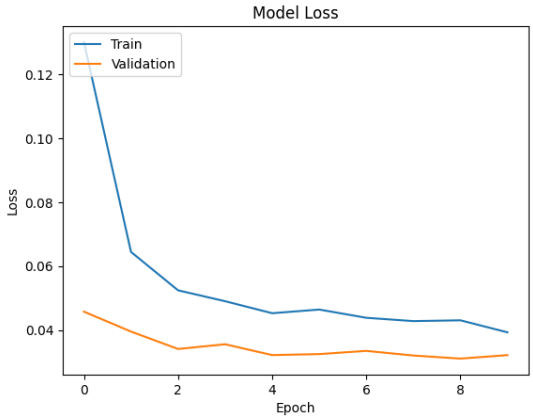

Figure 5: Model Loss

The Figure. 6, confusion matrix measures the efficiency of a classification algorithm relative to five classes (0–4). Indeed, most of the predictions reflect the true labels, which can be seen by off-diagonal elements (54,242 for Class 0 and 54,415 for Class 1). There are very few

misclassifications, for example, 792 samples misclassified to Class 1 instead of True Label 3 and 52 samples to Class 4 instead of True Label 3. The off-diagonal values crossing the diagonal are low and this proves the capability of the model in correct classification and the matrix also suggests the areas to be improved with identifying the right misclassifications.

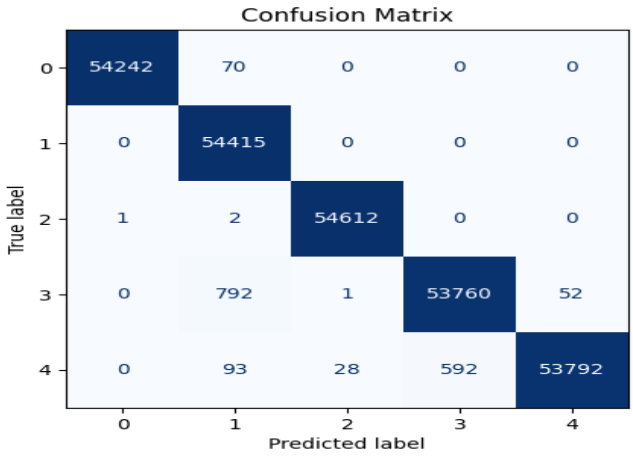

Figure 6: Confusion Matrix

The ROC-AUC graph of the multi-class also classifies the specimens according to the performance of the classification tests in five classes (0–4). Both classes obtain confusion matrix values and Area Under the Curve (AUC) with 1.00, representing remarkable classification performance. This curves also show a very good trade off between the true positive rate and false positive for all classes, indicating the good ability of the model in class separation. It is depicted in Figure 7,

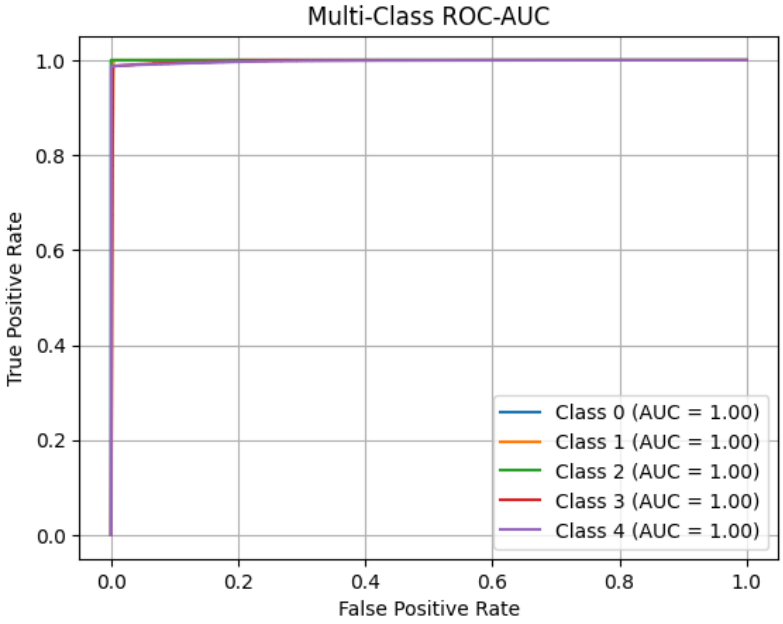

Figure 7: ROC-AUC Graph

The Figure 8, shows the curve of Precision with Recall measures the configuration effectiveness for five classes of 0–4. The receiver operating characteristics (ROC) are close to ideal, Class 0, Class 2, Class 4 having an AUC = 1, Class 1 and Class 3 having an AUC = 0.99. These curves reveal the high precision maintained by the model when the recall is varied, which is a great degree of classification in between classes. As seen from the nearly flat curves, the experiments give high classification accuracy, with little to no price in terms of precision and recall.

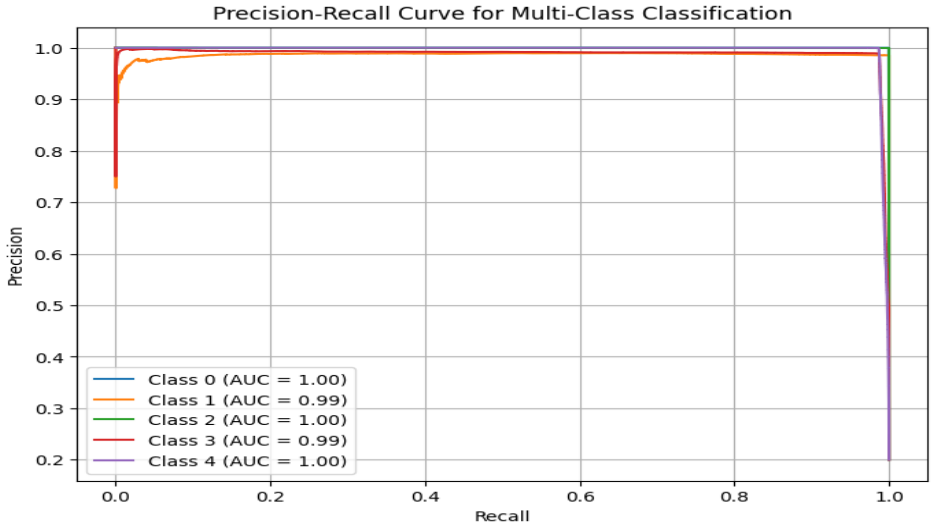

Figure 8: Precision- Recall Graph

The Figure 9, shown below is called F1-Score by Class which denotes the F1-scores of five classes (0 to 4). All classes have a perfect score represented by the value of F1-score equals to 1. This means that the study shows that the classification model has a good precision and recall ratio for all the classes. An ideal F1-score of all classes indicates the model has a high level of accuracy, moreover, the model generalizes well to the dataset.

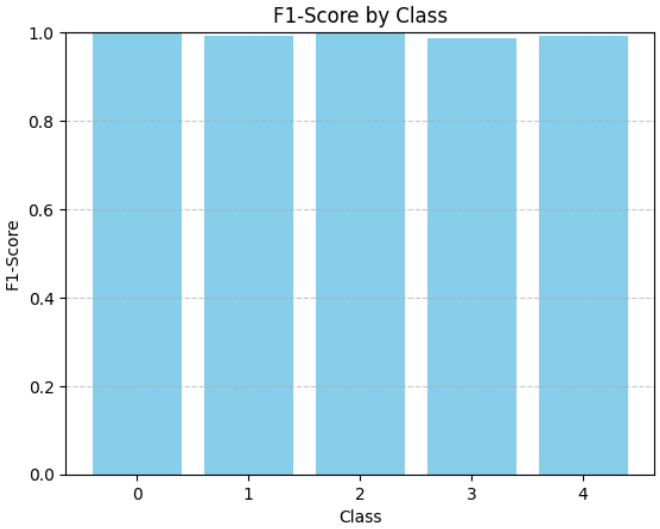

Figure 9: F1-ScoreGraph

The Figure 10, shows "Cumulative Gains Chart" analyses the model's performance in the correct placement of the positive cases. The x-axis is created with the amount of samples use in the successive runs while the y-axis reflects the proportion of positive samples achieved. The chart includes two lines: The two metrics include; the Cumulative Gain Curve (blue) that climbs sharply to the right suggesting that in the early ranks the model identifies the majority of positive cases and hence is positive and the Baseline (Red dashed line) implying on the other end that it is a random model. The fact that blue curve rises steeply above base line indicates the model has high positive predicting ability, meaning, the model is far better than random selection of positive instances.

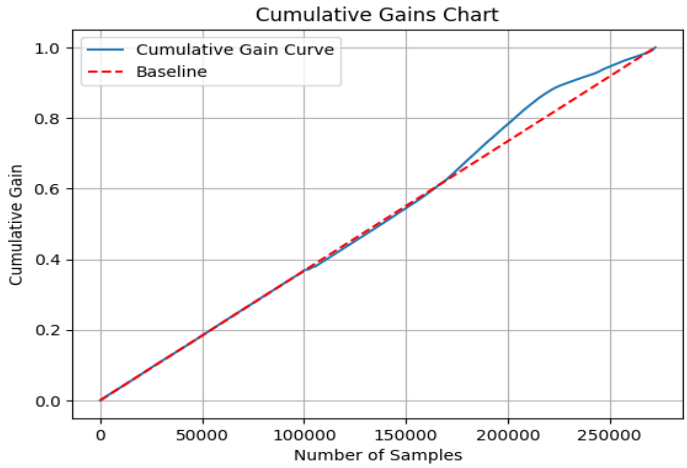

Figure 10: Cumulative Gains Chart

The Table 1, shows the evaluation of model performances shows that the accuracy of the Proposed MLP-GRU model is the highest, 99.4%. Classification report shows that Support Vector Machine (SVM) leads by acquiring a 99.32% accuracy score, followed by Decision Tree (DT) with a 99%, and last is CNN with 98.76%. The SVM and DRNN methods presents little lower accuracy of 96.41% and 96% respectively. Figure 11, depicts the MLP-GRU model proposed here is found to attain higher classification accuracy than other models.

6.2 Performance Evaluation

- **Accuracy:** Accuracy is a measure of how many predictions a particular model got right in total forecasts. It is computed using the following equation (1). Hence, the higher the accuracy value, the better positioned the model is in properly predicting results. However, the most accurate in case of balanced data because it may sometimes be deceptive in case of imbalanced data**.**

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ Predictions} \times 100 \qquad (1)$$

Table 1 : Performance Comparison of the Proposed Model

Method Name	Accuracy
DT	99%
CNN	98.76%
SVM	96.41%
DRNN	96%
ProposedMLP-GRU model	99.4 %

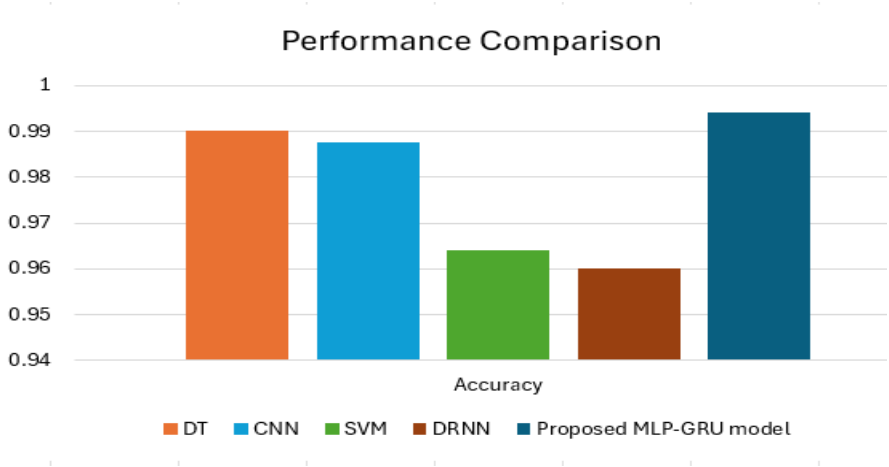

Figure 11: Performance Comparison

6.3 Discussion

The findings place the proposed MLP-GRU model in a position of offering a great improvement on the existing classification performance on imbalanced datasets. In contrast, MLP-GRU achieved superior performance to DT, CNN, SVM and DRNN and had the best accuracy of 99.4%. This proves the fact that even in situation of class imbalance it performs very efficient generalization and the precision / recall value is excellent. The outcome observed in the study is consistent with the reference benchmark and in terms of predictive accuracy achieved is better than the previously published work, along with lesser time complexity. The purpose of this work was to enhance the accuracy of multi-class classification for datasets which contain imbalanced class distributions. Downsampling such as class balancing was carried out and evaluation done by PR curves, F1 scores and ROC AUC on such instances as class balancing. However, for example, such as increasing the computational cost of the model, improving interpretability and many more challenges. However, the proposed MLP-GRU model can be easily scaled up to other ML

problems include real time predictions, and dynamic data sets which can be taken as a disadvantage but highly recommended for further research. Consequently, based on the results obtained, it can be stated that the specified model will be effective in relation to the suggested dataset with complex structural pattern with a view of optimizing the classification accuracy.

7 Conclusion and Future Work

The findings of this work show the effectiveness of the presented MLP- GRU model in increasing classification efficiency of imbalanced datasets. When compared to DT, CNN and SVM as well as DRNN, the MLP-GRU model proposed in this paper showed better performance of accuracy of 99.4%. This illustrates how it can generalize and remain in given levels of precision and recall of the course of evaluation, even at the occurrence of class imbalances. The presented results are to a large extent consistent with previous benchmarks and improve the results of the prior studies in terms of both accuracy and complexity of computations. The goal of this study was to achieve high prediction performance of the models in the case of multi-class classification and overcoming the difficulties that arise with the data imbalance. Downsampling were applied to bring the class distribution into proportion and target measures including precision-recall, F1 scores, and ROC-AUC were measures of performance. Nevertheless, issues like, a balance of the model's computation expense and improving explain ability are still a problem. However, this study's scalability to other domains which include real-time shortcut detection and dynamic datasets has not been explored. In general, the proposed MLP-GRU model indicates promising enhancements in classification accuracy, but it is about the computational efficiency and other common machine learning problems that need further study.

References

Acharya, J. *et al.* (2021) 'Detecting Malware, Malicious URLs and Virus Using Machine Learning and Signature Matching', in *2021 2nd International Conference for Emerging Technology (INCET)*. *2021 2nd International Conference for Emerging Technology (INCET)*, pp. 1–5. Available at: https://doi.org/10.1109/INCET51464.2021.9456440.

Akhtar, M.S. and Feng, T. (2022) 'Malware Analysis and Detection Using Machine Learning Algorithms', *Symmetry*, 14(11), p. 2304. Available at: https://doi.org/10.3390/sym14112304.

Albakri, A. *et al.* (2023) 'Metaheuristics with Deep Learning Model for Cybersecurity and Android Malware Detection and Classification', *Applied Sciences*, 13(4), p. 2172. Available at: https://doi.org/10.3390/app13042172.

Alomari, E.S. *et al.* (2023) 'Malware Detection Using Deep Learning and Correlation-Based Feature Selection', *Symmetry*, 15(1), p. 123. Available at: https://doi.org/10.3390/sym15010123.

Hakim, S.B. *et al.* (2024) 'Decoding Android Malware with a Fraction of Features: An Attention-Enhanced MLP-SVM Approach'. arXiv. Available at: http://arxiv.org/abs/2409.19234 (Accessed: 13 November 2024).

Hossain, Md.A. and Islam, Md.S. (2024) 'Enhanced detection of obfuscated malware in memory dumps: a machine learning approach for advanced cybersecurity', *Cybersecurity*, 7(1), p. 16. Available at: https://doi.org/10.1186/s42400-024-00205-z.

Jeon, S. and Moon, J. (2020) 'Malware-Detection Method with a Convolutional Recurrent Neural Network Using Opcode Sequences', *Information Sciences*, 535, pp. 1–15. Available at: https://doi.org/10.1016/j.ins.2020.05.026.

Jha, S. *et al.* (2020) 'Recurrent neural network for detecting malware', *Computers & Security*, 99, p. 102037. Available at: https://doi.org/10.1016/j.cose.2020.102037.

John Oluwafemi Ogun (2024) 'Advancements in automated malware analysis: evaluating the efficacy of open-source tools in detecting and mitigating emerging malware threats to US businesses', *International Journal of Science and Research Archive*, 12(2), pp. 1958–1964. Available at: https://doi.org/10.30574/ijsra.2024.12.2.1488.

M., G. and Sethuraman, S.C. (2023) 'A comprehensive survey on deep learning based malware detection techniques', *Computer Science Review*, 47, p. 100529. Available at: https://doi.org/10.1016/j.cosrev.2022.100529.

Narasimha Rao, K.P. and Chinnaiyan, S. (2024) 'Blockchain-Powered Patient-Centric Access Control with MIDC AES-256 Encryption for Enhanced Healthcare Data Security', *Acta Informatica Pragensia*, 13(3), pp. 374–394. Available at: https://doi.org/10.18267/j.aip.242.

Oliveira, A.S. de and Sassi, R.J. (2023) 'Behavioral Malware Detection Using Deep Graph Convolutional Neural Networks'. Available at: https://www.authorea.com/doi/full/10.36227/techrxiv.10043099.v1?commit=664084a4beae5e3c0edbcdb24a022e15e2ae16a3 (Accessed: 16 November 2024)
.
Ozkan-Okay, M. *et al.* (2024) 'A Comprehensive Survey: Evaluating the Efficiency of Artificial Intelligence and Machine Learning Techniques on Cyber Security Solutions', *IEEE Access*, 12, pp. 12229–12256. Available at: https://doi.org/10.1109/ACCESS.2024.3355547.

Puneeth, R.P. and Parthasarathy, G. (2024) 'Blockchain-Based Framework for Privacy Preservation and Securing EHR with Patient-Centric Access Control', *Acta Informatica Pragensia*, 13(1), pp. 1–23. Available at: https://doi.org/10.18267/j.aip.225.

Qureshi, S.U. *et al.* (2024) 'Systematic review of deep learning solutions for malware detection and forensic analysis in IoT', *Journal of King Saud University - Computer and Information Sciences*, 36(8), p. 102164. Available at: https://doi.org/10.1016/j.jksuci.2024.102164.

Singh, Jagsir and Singh, Jaswinder (2020) 'Detection of malicious software by analyzing the behavioral artifacts using machine learning algorithms', *Information and Software Technology*, 121, p. 106273. Available at: https://doi.org/10.1016/j.infsof.2020.106273.

Som, S. *et al.* (2021) 'Utilizing Gated Recurrent Units to Retain Long Term Dependencies with Recurrent Neural Network in Text Classification', *Journal of Information Systems and Telecommunication*, 9(34), pp. 89–102. Available at: https://doi.org/10.52547/jist.9.34.89.

Wu et al. (2024) 'Patient-centric medical service matching with fine-grained access control and dynamic user management', *Computer Standards & Interfaces*, 89, p. 103833. Available at: https://doi.org/10.1016/j.csi.2024.103833.

Zhao, Z. *et al.* (2023) 'Image-Based Malware Classification Method with the AlexNet Convolutional Neural Network Model', *Security and Communication Networks*, 2023(1), p. 6390023. Available at: https://doi.org/10.1155/2023/6390023.

Lu, T., Du, Y., Ouyang, L., Chen, Q. and Wang, X. (2020). Android Malware Detection Based on a Hybrid Deep Learning Model. Security and Communication Networks, 2020, pp.1–11.doi:https://doi.org/10.1155/2020/8863617.

Rimon, S.I. and Haque, Md.M. (2022). Malware Detection and Classification Using HybridMachine Learning Algorithm. Springer eBooks, pp.419–428. doi:https://doi.org/10.1007/978-3-031-19958-5_39.

R. Chaganti, V. Ravi, and T. D. Pham, "Deep learning based cross architecture internet of things malware detection and classification," Comput. Secur., vol. 120, p. 102779, Sep. 2022, doi: 10.1016/j.cose.2022.102779.

I. Ullah and Q. H. Mahmoud, “An Anomaly Detection Model for IoT Networks based on Flow and Flag Features using a Feed-Forward Neural Network,” in 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), Jan. 2022, pp. 363–368. doi: 10.1109/CCNC49033.2022.9700597.

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

Taheri, R., Ghahramani, M., Javidan, R., Shojafar, M., Pooranian, Z. and Conti, M., 2020. Similarity-based Android malware detection using Hamming distance of static binary features. Future Generation Computer Systems, 105, pp.230-247

P. Yang, G. Zhao, and P. Zeng, “Phishing Website Detection Based on Multidimensional Features Driven by Deep Learning,” IEEE Access, vol. 7, pp. 15196–15209, 2019, doi: 10.1109/ACCESS.2019.2892066.

Mustafa Majid, A.-A., Alshaibi, A. J., Kostyuchenko, E. & Shelupanov, A., 2021. A review of artificial intelligence based malware detection using deep learning. s.l., s.n.

Hadiprakoso, R., Buana, I. and Pramadi, Y., 2020. Android Malware Detection Using Hybrid-Based Analysis & Deep Neural Network. 2020 3rd International Conference on Information and Communications Technology (ICOIACT),.