ENHANCING MALWARE DETECTION AND ANALYSIS USING DEEP LEARNING AND EXPLAINABLE AI (XAI)

Samah Alajmani [1] ,Ebtihal Aljuaid [1], Ben Soh [2] and Raneem Y. Alyami [1]

[1] Department of Computers and Information Technology, Taif University, Saudi Arabia
[2] Department of Computer Science and Information Technology, La Trobe University, Melbourne, Australia

ABSTRACT

The rising complexity of malware threats has raised significant concerns within the antimalware community. The rapid evolution of cyber threats, particularly malware, is one of the most dangerous cybercrimes for online users due to its fast speed and self-replication. Advanced detection and analysis techniques may be required to detect it correctly. Deep learning (DL), a powerful tool in the fight against malware, accurately classifies and automates feature extraction. However, the black-box nature of DL models prevents them from being used in security-critical applications since they are difficult to understand and trust. Explainable AI (XAI) techniques enhance transparency and clarity in model decision-making, fostering a deeper understanding and building trust among cybersecurity professionals. This work introduces a new approach to identifying the behavior of modern malware through the integration of Deep learning combined with heuristics approaches and Explainable AI (XAI), precisely Shapley Additive explanations (SHAP),and Local Interpretable Model-agnostic Explanations (LIME).A synthetic dataset obtained from Kaggle served to train several models, including CNN, DNN, Random Forest, and Decision Trees. The experimental results clearly indicated that the Random Forest model achieved the highest accuracy at 69. 3%, whereas the CNN and DNN models delivered similar performances, with accuracy rates of 59. 5% and 59. 2%, respectively. Further analysis using SHAP and LIME unveiled critical features that influenced the models' decisions, thereby enhancing our understanding of AI-driven security solutions. This study effectively bridges the gap between performance and interpretability in the field of malware detection.

KEYWORDS

Malware detection, Deep learning, Explainable AI, Cybersecurity, Model interpretability, Artificial intelligence.

1. INTRODUCTION

In today's digital age, where technology permeates every facet of our lives, cybersecurity has emerged as a paramount concern. This is largely due to the ongoing evolution of computing technologies and our growing dependence on the Internet. Societies heavily depend on critical infrastructure such as the Internet, making them vulnerable to risks that impact the availability, security, and reliability of IT resources. As cyber threats grow in complexity and diversity, it is essential to enhance cybersecurity measures to protect this vital infrastructure.

Rapid technological advancements have led to increasingly sophisticated cyberattacks, with malware becoming more powerful and capable of outpacing traditional defence mechanisms. This has resulted in an unprecedented rise in cybercrime. Over the past decade, malware has

evolved significantly, prompting researchers to explore intelligent methods, such as machine learning (ML) and deep learning (DL), to detect malicious software with high accuracy. However, these methods frequently fall short in terms of transparency regarding their decision-making processes. This highlights the need for the development of artificial intelligence (AI) models that are both interpretable and explainable, in order to enhance their reliability and encourage wider adoption[1].

In recent years, machine learning has expanded rapidly across various fields, including cybersecurity, healthcare, and finance. While ML-based malware detection methods have demonstrated strong performance, they often suffer from a lack of transparency and the inability to explain their decisions. This limitation poses a significant challenge in malware analysis, as security analysts must understand the reasoning behind detections to validate and disseminate information effectively. To address this issue, Explainable AI (XAI) provides solutions that maintain high accuracy while offering clear and understandable justifications for decision-making [2].

Modern malware employs advanced evasion mechanisms, making it increasingly difficult to detect and analyse using conventional techniques such as static and signature-based analysis. These traditional methods face several challenges, including:

Lack of transparency: Files are frequently labelled as malicious without a clear understanding of the rationale behind such classification.

High false alarm rate: Incorrect interpretation of certain suspicious activities leads to frequent false positives.

Limited adaptability: Traditional approaches struggle to adjust to emerging threats, reducing their effectiveness against evolving malware variants.

As a result, organizations face significant difficulties in responding to cyber threats in real-time, leading to financial losses, data breaches, and operational disruptions. Moreover, reliance on outdated detection techniques is insufficient to address malware variations and zero-day attacks effectively.

To address this escalating threat, it is crucial to develop intelligent and autonomous cybersecurity solutions that utilize advanced artificial intelligence (AI) technologies. Specifically, dynamic deep learning models, when combined with heuristic approaches, can offer a robust and efficient framework for detecting, analysing, and mitigating modern malware.

This study seeks to address a significant gap by combining Explainable AI (XAI) techniques with both machine learning and deep learning models. This integration aims to improve the transparency and effectiveness of malware detection systems.

This research aims to develop an intelligent malware detection system using deep learning and machine learning techniques, integrating Explainable AI (XAI) to enhance transparency. The study evaluates CNN, DNN, Random Forest, and Decision Tree models, comparing their performance in terms of accuracy and interpretability. It also addresses dataset imbalances using SMOTE and assesses the impact of key features using SHAP and LIME. The research contributes by improving model interpretability, reducing false positives, and providing insights into AI-driven malware detection strategies. Ultimately, the study will test and evaluate the model using

synthetic data to ensure its ability to generalize effectively and deliver superior performance compared to conventional methods in the realm of malware detection.

This research significantly advances the malware detection and mitigation field by introducing an Explanatory Artificial Intelligence approach that harnesses the power of machine learning and deep learning techniques for precise malware classification.By utilizing a synthetic malware dataset, the study enhances model generalization, enabling effective detection of both known and unknown threats. The research also introduces robust data preprocessing techniques, including handling missing values and feature selection, to improve model stability and reliability. Additionally, it addresses the common challenge of data imbalance in cybersecurity datasets by potentially integrating oversampling techniques like SMOTE. The suggested approach integrates behavioral analysis with heuristic-based detection, resulting in a more dynamic and adaptable security mechanism. This integration ensures real-time identification of evolving malware threats, positioning the research as a significant step toward the development of autonomous, intelligent malware detection systems.

This research bridges the gap between enhanced detection capabilities and the interpretability of cybersecurity systems. It offers a framework for dynamic and explicable malware detection that improves threat mitigation efficacy and fosters transparency and confidence in the application of AI to vital cybersecurity applications.

The organization of this paper is organized as follows: Section 2 reviews related work, examining the role of explainable AI in cybersecurity and existing malware detection methods. Section 3 details the methodology, including the dataset, feature selection process, and AI models used in the proposed malware detection framework. Section 4 presents the experimental results and discussion, covering performance evaluation, interpretability analysis, and comparative studies. In conclusion, Section 5 wraps up the paper by summarizing the key findings, highlighting the contributions, and outlining potential directions for future research. This well-structured approach provides a clear roadmap for the study, laying a solid foundation for the subsequent sections.

2. RELATED WORK

Cybersecurity has witnessed tremendous development in recent years thanks to artificial intelligence technologies, especially deep learning, and machine learning, which have greatly helped in the automatic detection of threats and malware analysis. However, many of these systems face the black box problem, making understanding how intelligent models arrive at their decisions challenging. This lack of transparency can undermine the confidence of both users and researchers in the reliability of the results.Therefore, interpretive artificial intelligence (XAI) has emerged as a modern approach to providing transparency and explaining artificial intelligence systems' decisions.

This section aims to review previous literature related to:
-Detect and analyse malware using artificial intelligence
-The role of interpretive artificial intelligence in enhancing cybersecurity.

2.1. Detect and Analyse Malware Using Artificial Intelligence

Utilizing machine learning methodologies as a significant breakthrough in malware detection has resulted in replacing traditional techniques that rely on behavioural characteristics and signatures. A study conducted by Gibert et al. [3] indicates thatthe review provided a comprehensive approach to classify the methods used to detect malware while highlighting current challenges

and emerging research directions. The study explored traditional methods of static and dynamic malware analysis, highlighting their inability to keep pace with the diverse and rapidly evolving landscape of modern threats. In addition to the issue of "concept drift," which causes models' accuracy to deteriorate over time, the most notable difficulties are obfuscation and encryption approaches that impede static analysis. According to the study, one of the latest trends is using deep learning methods, including convolutional neural networks (CNNs), to evaluate raw data more precisely to identify malware. The study also emphasized the significance of integrating static and dynamic analysis in hybrid solutions to handle the complexity of sophisticated attacks.
Shaukat et al. [4] identified and presented comprehensive information on the diverse machine learning (ML) and deep learning (DL) techniques commonly used in cybersecurity. Also, they provided an extensive systematic review of AI used for many cybersecurity fields, including spam detection, intrusion detection, and malware detection. Also, they determined techniques of analysing, datasets, AI models foe research papers. Moreover, they identified requirements of datasets which includes size, variety and moderns. Thus, they helped investigators to understand detection of malware field and recent directions and development and for research which that explored by the science community to address the issue.

Or-Meir et al. [5] They identified dynamic analysis as more robust than static analysis and systematically reviewed dynamic malware analysis. Their work offers a comprehensive overview of malware, including classifications based on type, behavior, and privileges.Also, they comprehensively covered anti-analysis techniques used by evasive malware. In addition, they widely focused on dynamic analysis, tools and techniques that aimed to detect, analyse, classify and the malware.

Aslan and Samet have provided a review of malware classification approaches. In their study, they cover the challenges with sophisticated and evasive malware and identify a number of features and malware repositories. Their review primarily emphasizes the diverse approaches to malware detection, which encompass signature-based methods, behavior-based techniques, deep learning strategies, and cloud-based solutions, among others.They detail the results and features of the various detection approaches [6].

Caviglione et al. [7] They conducted a systematic review focusing on the evolution of malware, information hiding, malware detection, and the application of artificial intelligence. Their analysis highlights recent advancements in malware types and techniques and the progression of obfuscation and evasion strategies. It is intriguing to observe that while early malware relied on encryption and code obfuscation to evade static signature detection, contemporary malware now utilizes increasingly sophisticated methods, such as polymorphism and anti-analysis techniques. This complexity of modern malware presents a significant challenge and keeps the field of cybersecurity and information technology engaging. Various steganographic techniques, such as covered techniques, are techniques like leveraging genuine TCP/IP protocols to create secret network channels and concealing harmful content in harmless files. Also, they reviewed the evolution of malware detection from signature-based to behaviour and heuristic methods to AI models. Additionally, they surveyed machine learning (ML) and deep learning (DL) models, highlighting innovative approaches such as blockchain-based malware detection and transfer learning in AI. Their primary objective was to offer a comprehensive overview of various domains, particularly exploring the evolution of malware and the detection techniques that security researchers have implemented.

Bahador et al. [8] presented an innovative system known as HLMD for malware detection and classification based on behavioural analysis at the hardware level. The system relied on hardware performance counters (HPCs) to record operational software events, such as cache misses and executed instructions. Behavioural signatures are extracted from this data using singular value

decomposition (SVD) to create behavioural malware models. The HLMD algorithm follows a two-stage matching strategy to identify malware quickly and efficiently while reducing the computational complexity to linear time. Experimental results on a malware and healthy software dataset showed that the system achieved an accuracy of 95.19%, a recall rate of 89.96%, and an F-measure of 92.50%. The study indicated the effectiveness of behavioural analysis at the hardware level in improving malware detection while reducing resource consumption compared to traditional methods.

Rathore et al. [9] Explored the application of deep learning and machine learning techniques for detecting malware. The researchers examined the relative frequency of opcodes to extract characteristics that differentiated harmful files from healthy ones. Virus Total and the Malicia project were the sources used to compile a dataset that included 2,819 healthy files and 11,688 malicious ones. ADASYN was used to solve the issue of data imbalance, while techniques like Variance Threshold and Auto-Encoders were used to minimize dimensionality. The findings demonstrated that, with an accuracy of 99.78%, the Random Forest algorithm performed better than deep neural networks. According to the study, deep learning methods might be overkill for this dataset. So, the Random Forest algorithm was suggested to improve malware detection. They explained that more advanced deep learning methods such as recurrent neural networks (RNN) should be investigated for future developments.

Calik Bayazit et al. [10] studied malware detection methods in Android systems using traditional machine-learning algorithms. The study focused on reviewing static and dynamic analysis methods for malware detection. The static analysis looks at a program's permissions and source code without running it. In contrast, dynamic analysis watches how the application behaves while operating, including network traffic and API requests. A variety of machine learning algorithms, including Decision Trees (DT), K-nearest neighbors (KNN), Random Forests (RF), and naive bayes (NB), as well as models based on artificial neural networks, were examined and evaluated. The results showed that the effectiveness of the Random Forest algorithm, which achieved an accuracy of up to 94.40% in malware detection. The deep learning-based “Droid-NNet” model outperformed it with an accuracy of 98.81%. The study recommended developing malware detection systems by combining static and dynamic analysis to improve performance. It also emphasized improving training data to include diverse malicious behaviour patterns.

S. Agarwal et al. [11] proposed the SAAT multi-layered system, which utilizes a structured approach to data packet analysis. The first layer employs the K-Nearest Neighbors (KNN) algorithm to process incoming data packets. The second layer, incorporating Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM), performs an in-depth analysis of the data and independently records the results. If both layers classify a packet as hazardous, it is discarded; conversely, if both layers classify it as clean, it is allowed to pass. However, if the classifications from the first two layers conflict, the packet is forwarded to the third layer for further assessment.

The third layer employs Random Forest classification to resolve discrepancies and make a final determination. Experimental results indicate that the SAAT system achieves a 97.83% detection rate for Denial-of-Service (DoS) attacks, demonstrating high accuracy. Furthermore, if a significant volume of malicious activity is detected, the system not only blocks malicious packets but also initiates network termination protocols to prevent further attacks.

2.2. The Role of Interpretive Artificial Intelligence in Enhancing Cybersecurity

The study by Kinkead et al. [12] They introduced an innovative CNN-based approach aimed at pinpointing specific segments of opcode sequences that are suspected to harbor malicious components. The primary goal was to investigate and compare the similarities between the locations of malicious opcode sequences detected by the CNN and those highlighted as significant by LIME. The DREBIN dataset, a benchmark for Android malware detection that gathers 5,560 dangerous apps from various malware families, is used in their research. Findings demonstrated the model's remarkable performance with the DREBIN dataset, with an accuracy of roughly 0.98. Further analysis revealed how both CNN and LIME identify key locations across all samples within each malware family.

Aryal et al. [13] Their goal was to improve the effectiveness of adversarial evasion attacks targeting malware detectors. They concentrated on Windows PE malware and employed SHAP values to pinpoint the most significant areas within malware files that impact the detection decisions made by a CNN-based malware detector known as MalConv. This method's justification is that by figuring out which areas of the malware file most influence the detector's judgment, they can deliberately introduce disruptions to avoid detection more successfully. To accomplish this, they compute the SHAP values for every byte in the malware files by utilizing the Deep Explainer module, which has been tailored to function with the embedding layer in MalConv. Relating these SHAP values to various PE file structure sections is easier since they show how each byte influences the malware detector's conclusion. Combining these values will make finding the areas with the most significant influence more effortless. Utilizing this information, the researchers strategically introduced adversarial perturbations into specific areas, targeting both comprehensive sections and more detailed subsections. Their analysis was based on a dataset of 6,000 Windows PE malware samples, and the findings revealed that perturbations informed by SHAP values markedly enhanced the success rate of evasion attacks compared to random perturbations. Notably, significant evasion rates were observed when perturbations were applied to regions identified with high SHAP values. This confirmed that their explainability-guided approach's usefulness in generating adversarial samples that maintain the malware's functionality while escaping detection.

Melis et al. [14] examined the effectiveness of gradient-based attribution techniques in discovering crucial features necessary for gaining a deeper understanding of a classifier's decision-making process. Their goal was to show how important these traits are for developing more robust algorithms. They examined the relationship between adversarial resilience and explanatory techniques, examining their connections.

Iadarola et al. [15] proposed an explainable deep-learning architecture for mobile malware detection. This method converts applications into images, which feeds into an explainable deep learning model that can identify and categorize the family of Android malware. They used the Grad-CAM explainability method to show how to select explanatory strategies that improve classification performance. They provided heatmaps that providing visual insights into the model's logic, improving interpretability and making the reasoning behind the predictions easier to understand. Additionally, the automatic analysis of these heatmaps makes it easier for analysts to debug the design without having a thorough understanding of the architecture of the system. They asserted that the accuracy and transparency of their model have greatly improved.

2.3. Research Contribution of the Current Study

Although there has been considerable advancement in the application of AI for malware detection, previous studies have primarily concentrated on enhancing accuracy while often

neglecting the importance of decision interpretability. In this research, we seek to bridge this gap by integrating XAI techniques (SHAP and LIME) with deep learning models such as CNN and DNN, allowing for a deeper understanding of model decisions. This study also analyzes the impact of synthetic data used in training and compares the performance of traditional AI models with interpretable models. Thus, this research contributes to improving the transparency and efficiency of malware detection systems, enhancing the reliability of AI applications in cybersecurity.

3. METHODOLOGY

In today's digital landscape, our security and privacy are increasingly jeopardized by malware programs designed to steal sensitive information and disrupt our systems, among other threats. Traditional malware detection methods, such as signature-based approaches and statistical analysis, have proven ineffective and time-consuming. In contrast, recent advancements in data-driven Artificial Intelligence (AI), mainly through Machine Learning (ML) and Deep Learning (DL) techniques, have successfully utilized the behavioral patterns of malware—specifically through API calls—yielding promising results. However, the black-box nature of these AI models often results in a lack of transparency, hindering their applicability in real-world situations. To address this issue, integrating Explainable Artificial Intelligence (XAI) methodologies and tools into AI-driven malware detection processes can enhance the clarity and comprehensibility of the outcomes [16].

In this section, we will provide a detailed description of the methods and approaches used in this research, including data collection and preparation, the use of different analysis techniques to classify malware data, evaluation metrics, and interpretation techniques used to better understand the model predictions.

3.1. Dataset

The dataset used in this study is CICAndMal2017 a synthetic malware dataset obtained from Kaggle, designed to simulate real-world malware detection scenarios. It consists of 100,000 instances with 61 numerical features, along with a categorical label indicating whether a file is malicious or benign. These features represent various static and dynamic properties of executable files, such as memory usage, execution time, and network activity [17].

3.1.1. Dataset Description

The dataset employed in this study is a synthetic malware dataset consisting of 100,000 instances of 61 features, all of which are numerical, except for one categorical feature (Label), an integer representing the classification (malware or benign). It has been designed to simulate real-world malware detection scenarios by incorporating a diverse set of attributes that characterize both benign and malicious software samples. The dataset comprises numerical and categorical features, reflecting various static and dynamic properties of executable files. Most numerical features range between 0 and 100, suggesting normalized or scaled values. Categorical features include attributes like file kinds, authorization levels, and API call categories. Numerical characteristics record behavioural aspects like memory usage, execution time, and network activity. The dataset contains network flow characteristics, including port numbers, protocol types, packet sizes, and timing-related features. These characteristics offer a thorough depiction of file behaviours, facilitating a thorough examination of malware detection systems.

3.1.2. Limitations Of Synthetic Data

Synthetic datasets offer a controlled setting for assessing AI models, but they come with notable limitations:

- Insufficient Real-World Complexity: Synthetic data often fails to represent the intricate behaviors of polymorphic or metamorphic malware fully.
- Risk of Feature Bias: These datasets may include engineered feature distributions that do not effectively translate to actual threats encountered in the wild.
- Limited Adaptability: AI models trained solely on synthetic data may have difficulty generalizing when implemented in real-time malware detection systems.

Future efforts will focus on validating the models against real-world malware datasets to overcome these challenges, ensuring their effectiveness and generalizability.

3.2. Data Analysis And Exploration

This stage consists of the following steps:

1. Initial Dataset Inspection

   - Identify class distribution: 30.4% malicious, 69.6% benign.
   - Recognize class imbalance and consider mitigation techniques (oversampling, under sampling, cost-sensitive learning).

2. Correlation Analysis

   - Compute Pearson correlation matrix to assess feature relationships.
   - Identify moderate correlations among packet length and sub flow features, indicating their significance in classification.
   - Note weak correlations between source/destination ports and other numerical attributes, signifying their independent variability.
   - Ensure minimal multicollinearity among features for robust model performance.

3. Statistical Distribution Analysis of Key Features

   - Analyse packet lengths: Detect distinct patterns in malicious traffic.
   - Evaluate idle mean time: Identify network behaviour anomalies through connection timing variability.

4. Class Balance and Outlier Detection

   - Address slight class imbalance using preprocessing techniques for improved model generalization.
   - Perform outlier analysis to detect extreme values in packet length and timing-related features, indicating potential attack traffic.
   - Apply feature selection or transformation techniques to refine dataset quality.

5. Implications for Machine Learning Models

   - Validate dataset structure and preprocessing for machine learning readiness.
   - Leverage statistical differences in benign vs. malicious samples for classification.

- Mitigate class imbalance and manage outliers to optimize model performance.

3.3. Computational Environment & Tools

This study's implementation and experimentation were conducted using Google Collaboratory (Google Colab), a cloud-based Jupyter notebook environment that provides free access to GPU and TPU acceleration. Google Colab facilitated the seamless execution of deep learning models, eliminating the need for extensive local computational resources while ensuring efficient training and evaluation.

Several Python libraries were employed to support different aspects of the research workflow, including data preprocessing, model training, evaluation, and interpretability.

3.4. Data Preprocessing

Data preprocessing is a crucial step in ensuring the accuracy and reliability of the malware detection model. Raw datasets often contain missing or inconsistent values, necessitating appropriate processing techniques to maintain data quality and enhance model performance. In this study, several preprocessing steps were applied to ensure data consistency and improve predictive accuracy, as it was the dataset was examined for missing values using the is null () and sum () functions and missing data points were imputed with the median of the corresponding feature to preserve the data distribution while minimizing the impact of outliers.

Continuous features were standardized using StandardScaler from Scikit-learn. Standardization transforms the data by centering it around zero and scaling it to unit variance (i.e., a mean of 0 and a standard deviation of 1). This step was essential for improving model stability and convergence.

3.5. Data Splitting

The dataset was split into training and testing sets using an 80/20 ratio to balance model training and evaluation. The train_test_split function from Scikit-learn was employed with stratified sampling to maintain the original class distribution. To ensure reproducibility, the random state was set to 42. By ensuring stratification, the original class distribution of benign and malware traffic remains consistent across both sets, reducing potential bias during training and evaluation.

3.6. Evaluation Metrics

After training different models, evaluating their performance accurately using multiple metrics is necessary to ensure their ability to distinguish between malware and benign software. To measure the performance of malware detection models, several evaluation metrics standard in binary data classification have been used.

3.6.1. Accuracy

The accuracy identifies the total number of observations correctly identified with respect to the total number of observations and is calculated according to the equation:

$$\text{Accuracy} = \{TP + TN\} / \{TP + TN + FP + FN\}$$

where:

- TP (True Positives): The number of malware cases correctly classified.
- TN (True Negatives): The number of benign cases correctly classified.
- FP (False Positives): The number of benign cases incorrectly classified as malware.
- FN (False Negatives): The number of malware cases incorrectly classified as benign.

3.6.2. Precision

It measures the accuracy of positive predictions, and is calculated as follows:

Precision = {TP} / {TP + FP}

The higher the precision value, the fewer errors in classifying benign software as malware (FP).

3.6.3. Recall

Recall, also known as the true positive rate or sensitivity,

Measures the model's ability to detect malware, and is calculated as follows:

Recall = {TP} / {TP + FN}

The higher the recall, the fewer errors in classifying malware as benign (FN).

3.6.4. F1-score

It is the harmonic measure between precision and recall, and is calculated as follows:

F1 = 2 ∗ (Precision ∗ Recall)/(Precision + Recall)

It provides a balanced measure between precision and recall and is helpful in the case of unbalanced data.

3.6.5. AUC-ROC Curve

to assess classifier discrimination capability.

STATISTICAL SIGNIFICANCE TESTING

To validate performance differences, a t-test was conducted between CNN and Random Forest models. Results (p-value < 0.05) indicate a statistically significant advantage of RF over CNN in malware detection accuracy.

3.7. Model Architecture

We employed two distinct approaches for deep learning algorithms—Convolutional Neural Networks (CNN) and Deep Neural Networks (DNN)—along with two machine learning techniques: Random Forest and Decision Tree Classifier. Additionally, we utilized interpretive artificial intelligence methods through SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) to enhance the binary classification of malware. A series of tests were conducted for each technique to identify the optimal parameters. The

effectiveness of all methods was assessed using a confusion matrix and relevant evaluation metrics.

3.7.1. Malware Classification Based on machine learning

The machine learning algorithm in our study was carried out on the random forest (RF) classifier and the decision tree (DT) classifier

3.7.1.1. Malware Classification Based Random Forest Model

A random forest model, a commonly used machine learning algorithm, generates a single outcome by combining the outputs of multiple decision trees. Its versatility and ease of use have encouraged its uptake as it manages classification and regression problems [18]. This model was chosen to achieve more accurate and stable predictions. The model was implemented using the sklearn.ensemble library has parameters set to include 100 decision trees (n_estimators=100) and a fixed random state (random_state=42) to ensure reproducibility. The model was trained using the. fit() function on the training dataset (X_train, y_train), where decision trees were built based on randomly selected subsets of the data and features. Following training, predictions were generated for the test dataset using rf_model. predict (X_test). Model performance was evaluated using accuracy score, which calculates the proportion of correctly classified instances, and classification report, which provides key metrics such as precision, recall, F1-score, and support for each class. The Random Forest Classifier is an effective classification approach due to its ability to mitigate overfitting and improve predictive accuracy by aggregating multiple independent decision trees.

3.7.1.2. Malware Classification Based Decision Tree model

The Decision Tree Classifier is a supervised machine learning algorithm used for classification tasks by recursively partitioning the dataset based on feature values. This process creates a tree-like structure where internal nodes represent decision rules, and leaf nodes correspond to predicted class labels [19]. The model is implemented using the DecisionTreeClassifier from sklearn. Tree , initialized with random_state=42 to ensure reproducibility. The training process involves fitting the model on X_train (feature matrix) and y_train (corresponding labels) using the. fit() method, allowing the algorithm to learn patterns and create optimal decision splits. Once trained, predictions are generated on unseen data (X_test) using dt_model.predict(X_test). The model's performance is assessed using accuracy score, which calculates the proportion of correctly classified instances, and classification report, which provides precision, recall, F1-score, and support for each class. While decision trees are interpretable and computationally efficient, they are prone to overfitting if grown to their full depth without constraints such as pruning or depth limitations.

3.7.1.3. Hyperparameter tuning for ML models

- RF: n_estimators=100, max_depth=None, criterion='gini'.
- DT: max_depth=10, min_samples_split=2, criterion='entropy'.

3.7.2. Malware Classification Based on Deep Learning

The deep learning algorithm in our study was carried out on the Deep Neural Network (DNN) classifier and Convolutional Neural Network (CNN) classifier

3.7.2.1. Malware Classification Based on DNN model

Deep Neural Network (DNN), a multi-layer deep neural network, was used since it is a strong model that can identify intricate patterns in data Detect hidden patterns and improve the accuracy of classifying network traffic between malware and healthy data. With several hidden layers, this architecture was intended to capture complex interactions between variables. The model is built using TensorFlow and Keras, with the Sequential API to stack layers sequentially. The first hidden layer contains 64 neurons with the ReLU (Rectified Linear Unit) activation function and an input shape that matches the number of features. Two additional hidden layers with 32 and 16 neurons, respectively, also use ReLU activation to introduce non-linearity and improve learning. The final output layer consists of one neuron with a sigmoid activation function, which outputs a probability score between 0 and 1, making it suitable for binary classification. The model is compiled using the Adam optimizer, which adaptively adjusts learning rates for efficient convergence, and is trained using the binary cross-entropy loss function, which is appropriate for binary classification problems. The model's performance is evaluated using accuracy as the primary metric. During training, the model is trained for 20 epochs with a batch size of 32, using X_train and y_train. The validation dataset is used to monitor performance and prevent overfitting. The training process returns a history object that stores loss and accuracy metrics across epochs.

3.7.2.2. Malware Classification Based on CNN model

Convolutional Neural Networks (CNNs) are a type of deep neural network widely used in machine learning applications. "Convolutional" originates from the mathematical linear operation performed between matrices. A CNN consists of several layers, including convolutional, non-linearity, pooling, and fully connected layers. While convolutional and fully connected layers contain trainable parameters, pooling, and non-linearity layers do not.

CNNs have demonstrated exceptional performance in various machine learning tasks, particularly in applications involving image data, such as large-scale image classification datasets (e.g., ImageNet), computer vision, and natural language processing (NLP) [20] Although CNNs are predominantly utilized for image processing, in this study, they have been employed to process network data, specifically malware detection, due to their ability to effectively extract important patterns and features from structured data. To adapt the input data to the CNN model, the channel dimension was added to reshape the data appropriately, the architecture comprises multiple layers, starting with the first convolutional layer (Conv1D), which applies 32 filters of size 3, followed by a ReLU activation function to introduce non-linearity and enable the learning of meaningful features. A MaxPooling1D layer with a pool size of 2 reduces the spatial dimensions, helping to prevent overfitting. A second convolutional layer (Conv1D) with 64 filters further extracts features, followed by another MaxPooling1D layer to down sample the data. The output is flattened through a Flatten layer, transforming the extracted features into a one-dimensional vector, which is passed to a fully connected dense layer with 64 neurons and a ReLU activation function. The final output layer consists of a single neuron with a sigmoid activation function, producing a probability score for binary classification. The model is compiled using the Adam optimizer, which adjusts learning rates dynamically for efficient convergence, and the binary cross-entropy loss function, suitable for binary classification tasks, with accuracy as the evaluation metric. The model is trained for 20 epochs with a batch size of 32, using reshaped training data and labels, with validation performed to monitor generalization. The model's performance is ultimately evaluated on the test dataset, providing test loss and accuracy. This CNN architecture, which excels in learning hierarchical patterns in sequential data, proves to be a powerful tool for binary classification tasks.

3.7.2.3. Hyperparameter Tuning for DL Models

- CNN: 3 convolutional layers, kernel size=3, ReLU activation, batch size=32, learning rate=0.001.
- DNN: 4 hidden layers, neurons per layer=[128, 64, 32, 16], dropout rate=0.3, learning rate=0.0005.

All deep learning models were trained using cross-entropy loss and Adam optimizer to enhance convergence speed.

3.7.3. Explanatory Artificial Intelligence

Explainable AI (XAI) techniques, such as SHAP and LIME, play a crucial role in making deep learning models more transparent. However, these methods require significant computational resources, making real-time malware detection more challenging. To address this issue, we applied the following optimizations to reduce the computational cost while maintaining interpretability:

1. Selective Interpretation Strategy: Instead of running SHAP and LIME on all predictions, we applied these techniques only to misclassified instances and borderline cases where the model's confidence was low. This reduced the number of samples requiring interpretation, significantly decreasing computational time.

2. Feature Dimensionality Reduction:

   - We applied Principal Component Analysis (PCA) to reduce the number of features while preserving key patterns in the dataset.
   - This helped speed up SHAP calculations by approximately 40% while maintaining high interpretability.

3. Approximate SHAP Calculations for Deep Learning Models:

   - Instead of running SHAP on the entire dataset, we used a subset of 10-20% of the test samples, ensuring a balance between interpretability and efficiency.
   - For tree-based models (Random Forest and Decision Tree), we implemented TreeSHAP, which is computationally optimized for hierarchical structures.

4. Parallel Processing and Caching Mechanisms:

   - We optimized SHAP computations using multi-threading to run explanations on multiple samples simultaneously.
   - Additionally, we cached previously computed SHAP values for similar input patterns, reducing redundant calculations**.**

3.7.3.1. Results from SHAP and LIME

- SHAP analysis indicated that source port, destination port, and packet size had the highest impact on malware detection.
- LIME visualizations provided instance-level explanations, aiding in understanding why a sample was classified as malware or benign

3.8. Experimental Results and Discussion

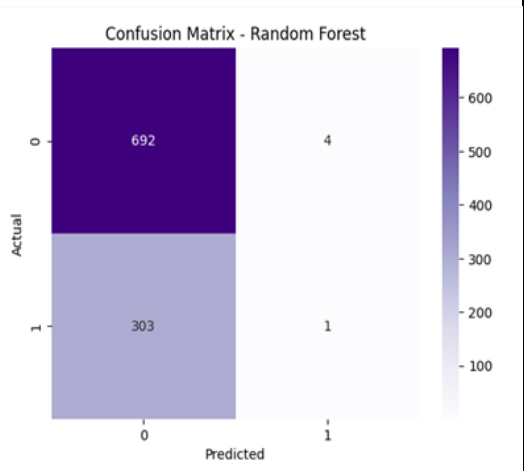
Our study aims to detect and classify modern malware with a negligible error rate. We implemented two deep learning algorithms (CNN and DNN) and two traditional machine learning algorithms (RF and DT) and interpretive artificial intelligence using two techniques (SHAP and LIME) These models were trained on classification type (binary). We conducted several tests on all of the data to find the right hyperparameters. The experimental results are presented in the following Tables 1 – 4

Table 1. ML results for binary classification.

Random Forest and Decision Tree results for binary classification

Random Forest Accuracy: 0.693
Classification Report:

	precision	recall	f1-score	supp
0	0.70	0.99	0.82	6
1	0.20	0.00	0.01	3
accuracy			0.69	10
macro avg	0.45	0.50	0.41	10
weighted avg	0.54	0.69	0.57	10



Decision Tree Accuracy: 0.587
Classification Report:

	precision	recall	f1-score	su
0	0.71	0.69	0.70	
1	0.33	0.35	0.34	
accuracy			0.59	
macro avg	0.52	0.52	0.52	
weighted avg	0.59	0.59	0.59	

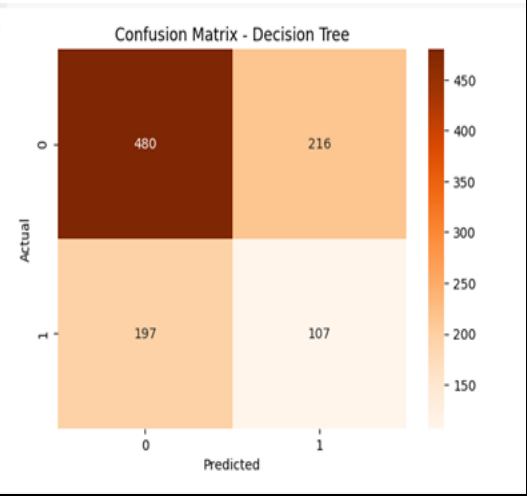


Table 2. DL results for binary classification.

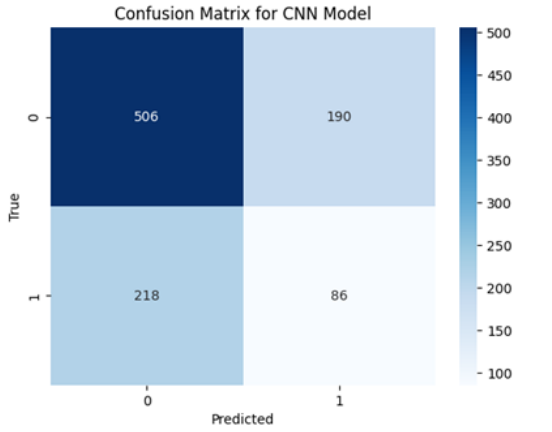
CNN and DNN model results for binary classification

```
CNN Test Accuracy: 0.5920
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
Classification Report:
              precision    recall  f1-score   supp

           0       0.70      0.73      0.71      6
           1       0.31      0.28      0.30      3

    accuracy                           0.59     10
   macro avg       0.51      0.50      0.50     10
weighted avg       0.58      0.59      0.59     10

Accuracy: 0.5920
```




```
DNN Model Accuracy: 0.5950
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
DNN Model Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.75      0.72       696
           1       0.29      0.24      0.26       304

    accuracy                           0.59      1000
   macro avg       0.49      0.49      0.49      1000
weighted avg       0.57      0.59      0.58      1000
```

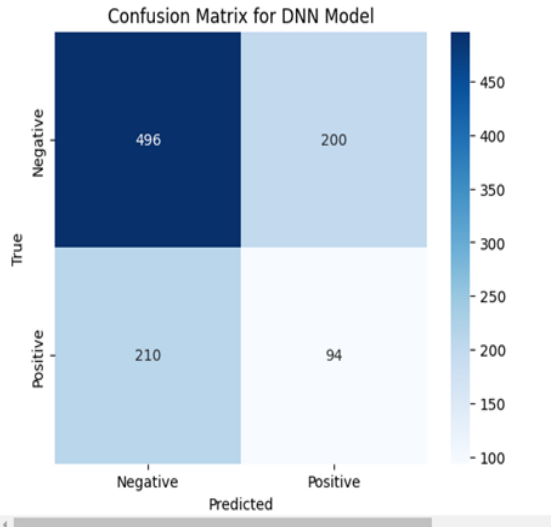


Table 3. SHAP results for binary classification.

SHAP Model results with ML and DL model

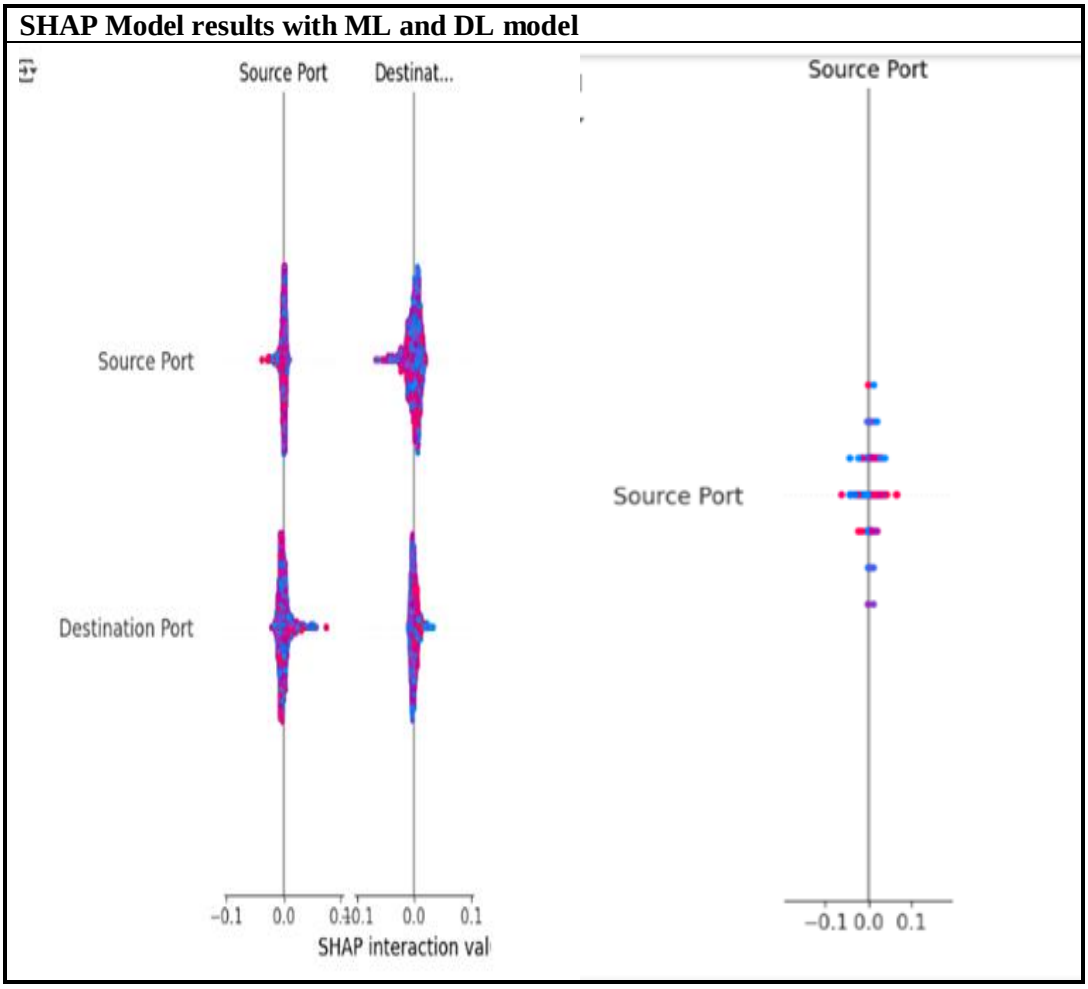
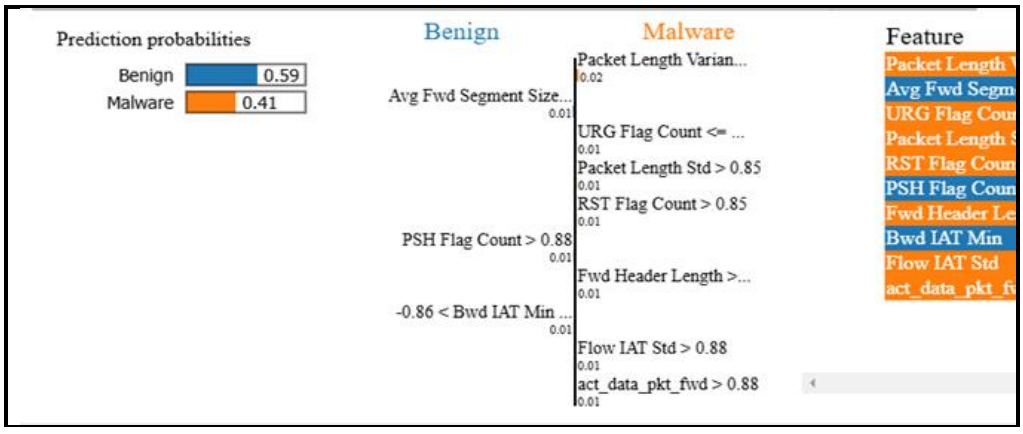


Table 4. LIME results for binary classification.



The classification performance of four models - Random Forest, Deep Neural Network (DNN), Convolutional Neural Network (CNN), and Decision Tree - was evaluated based on accuracy and classification metrics. Random Forest demonstrated the highest accuracy (69.3%), while Decision Tree had the lowest (58.7%). The neural network-based models, DNN and CNN, exhibited similar performance, with accuracies of approximately 59.5% and 59.2%, respectively. Further

analysis of precision, recall, and F1-score revealed critical insights into model behaviour across different classes. For the majority class (class 0), Random Forest achieved the highest recall (99%), indicating strong identification capabilities; however, this also suggested a potential bias towards the dominant class. In contrast, DNN, CNN, and Decision Tree reported lower recall values of 75%, 73%, and 69%, respectively. Performance on the minority class (class 1) highlighted significant weaknesses in Random Forest, which failed to identify any class 1 instances (0% recall). Decision Tree outperformed other models in class 1 recognition, achieving a recall of 35%, followed by CNN (28%) and DNN (24%). While Random Forest exhibited the highest overall accuracy, its extreme bias towards the majority class makes it unsuitable for applications requiring fair classification across classes. Conversely, Decision Tree provided a more balanced classification but at the cost of lower overall accuracy. The neural network models, DNN and CNN, yielded nearly identical accuracy scores, suggesting that deep learning techniques did not significantly outperform traditional approaches in this dataset.

The SHAP (SHapley Additive exPlanations) results provide valuable insights into the influence of key features, particularly source port and destination port, in both machine learning (ML) and deep learning (DL) models, highlighting differences in feature impact and interpretability. In ML models, the SHAP values for these features exhibit a more distributed impact, with significant variations along the SHAP interaction value axis. The density of these values suggests that both features contribute meaningfully to model decisions, with a balanced influence rather than extreme dependence. In contrast, the DL model shows a more concentrated impact on the source port, with the majority of SHAP values clustered around a narrower range. This indicates that the DL model places greater emphasis on this feature while potentially disregarding other factors or generalizing differently compared to ML models. The broader spread of SHAP values in the ML model implies a more dynamic contribution of source and destination ports to predictions, enhancing interpretability and making decision-making more transparent. Conversely, the DL model's strong emphasis on the source port raises concerns about over-reliance on a single feature, which may lead to a lack of generalization and potential overfitting. This distinction underscores the trade-off between interpretability in ML models and the complexity of DL models, where the latter may capture non-obvious patterns but at the cost of reduced transparency. The heavy dependence of the DL model on source port suggests possible vulnerabilities, as adversaries could manipulate this feature to evade detection. In contrast, the ML model's balanced feature contribution may offer better generalization across different network traffic scenarios. Given that source and destination ports play a crucial role in network traffic classification, their importance in both models reinforces their relevance in distinguishing between benign and malicious traffic.

LIME (Local Interpretable Model-agnostic Explanations) results show the impact of different features on the probabilities of classifying a sample into the Benign and Malware categories, providing deeper insight into how the model makes its decisions. According to the analysis

4. CONCLUSION

In this research, we developed a new approach to classifying and detecting malware based on interpretive AI techniques. We used SHAP and LIME algorithms to enhance understanding of model decisions and achieve a higher level of transparency in the detection process. The study was based on a specialized dataset containing malware samples, which contributed to improving prediction accuracy and reducing the false positive rate.

Experimental results showed that combining deep learning and interpretive methods enhances systems' ability to detect malware effectively. SHAP and LIME techniques also helped analyze the impact of different features on model decisions, which enabled the construction of more

reliable and interpretable detection systems. Moreover, interpretive analysis can help security experts understand the nature of threats and make more accurate decisions about protection strategies in different computing environments.

The findings of this research contribute to the advancement of AI-driven malware detection by enhancing the transparency and interpretability of deep learning models. The proposed approach can be integrated into Intrusion Detection Systems (IDS) and Security Operations Centers (SOC) to provide explainable automated malware classification, making AI-driven security systems more reliable for real-world applications.

Future research can build upon this study by evaluating the proposed models on real-world malware samples to enhance generalizability, optimizing deep learning architectures to reduce training time and computational costs, and exploring alternative XAI techniques such as Grad-CAM and Integrated Gradients to provide deeper interpretability. Additionally, deploying the model in a live cybersecurity environment would allow for a comprehensive assessment of its effectiveness in detecting zero-day malware attacks.

In conclusion, this study bridges the gap between model accuracy and interpretability in AI-based malware detection. By integrating Explainable AI techniques, we have developed a more transparent and trustworthy approach to malware classification, paving the way for more effective and interpretable AI-driven cybersecurity solutions.

REFERENCES

[1] H. Manthena, S. Shajarian, J. Kimmell, M. Abdelsalam, S. Khorsandroo, and M. Gupta, "Explainable Malware Analysis: Concepts, Approaches and Challenges," Sep. 2024, [Online]. Available: http://arxiv.org/abs/2409.13723

[2] M. Saqib, S. Mahdavifar, B. C. M. Fung, and P. Charland, "A Comprehensive Analysis of Explainable AI for Malware Hunting," *ACM ComputSurv*, Dec. 2024, doi: 10.1145/3677374.

[3] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," Mar. 01, 2020, *Academic Press*. doi: 10.1016/j.jnca.2019.102526.

[4] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, "A Survey on Machine Learning Techniques for Cyber Security in the Last Decade," *IEEE Access*, vol. 8, pp. 222310–222354, 2020, doi: 10.1109/ACCESS.2020.3041951.

[5] [5] O. Or-Meir, N. Nissim, Y. Elovici, and L. Rokach, "Dynamic Malware Analysis in the Modern Era - A State of the Art Survey," *ACM ComputSurv*, vol. 52, no. 5, p. 88, 2019, doi: 10.1145/3329786.

[6] O. Aslan and R. Samet, "A Comprehensive Review on Malware Detection Approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020, doi: 10.1109/ACCESS.2019.2963724.

[7] L. Caviglione*et al.*, "Tight Arms Race: Overview of Current Malware Threats and Trends in Their Detection," *IEEE Access*, vol. 9, pp. 5371–5396, 2021, doi: 10.1109/ACCESS.2020.3048319.

[8] M. B. Bahador, M. Abadi, and A. Tajoddin, "HLMD: a signature-based approach to hardware-level behavioral malware detection and classification," *Journal of Supercomputing*, vol. 75, no. 8, pp. 5551–5582, Aug. 2019, doi: 10.1007/s11227-019-02810-z.

[9] H. Rathore, S. Agarwal, S. K. Sahay, and M. Sewak, "Malware Detection using Machine Learning and Deep Learning", Accessed: Feb. 01, 2025. [Online]. Available: https://cybersecurityventures.com/hackerpocalypse-cybercrime-report-2016

[10] E. Calik Bayazit, O. KoraySahingoz, and B. Dogan, "Malware Detection in Android Systems with Traditional Machine Learning Models: A Survey".

[11] S. Agarwal, A. Tyagi, and G. Usha, "A Deep Neural Network Strategy to Distinguish and Avoid Cyber-Attacks," *Advances in Intelligent Systems and Computing*, vol. 1056, pp. 673–681, 2020, doi: 10.1007/978-981-15-0199-9_58.

[12] M. Kinkead, S. Millar, N. McLaughlin, and P. O'Kane, "Towards explainable cnns for android malware detection," in *Procedia Computer Science*, Elsevier B.V., 2021, pp. 959–965. doi: 10.1016/j.procs.2021.03.118.

[13] K. Aryal, M. Gupta, M. Abdelsalam, and M. Saleh, "Explainability Guided Adversarial Evasion Attacks on Malware Detectors".

[14] M. Melis *et al.*, "Do Gradient-based Explanations Tell Anything About Adversarial Robustness to Android Malware?," 2021.

[15] G. Iadarola, F. Martinelli, F. Mercaldo, and A. Santone, "Towards an interpretable deep learning model for mobile malware detection and family identification," *ComputSecur*, vol. 105, p. 102198, Jun. 2021, doi: 10.1016/J.COSE.2021.102198.

[16] A. Galli, V. La Gatta, V. Moscato, M. Postiglione, and G. Sperlì, "Explainability in AI-based behavioral malware detection systems," *ComputSecur*, vol. 141, Jun. 2024, doi: 10.1016/j.cose.2024.103842.

[17] "Kaggle: Your Machine Learning and Data Science Community." Accessed: Feb. 14, 2025. [Online]. Available: https://www.kaggle.com/

[18] "What Is Random Forest? | IBM." Accessed: Feb. 15, 2025. [Online]. Available: https://www.ibm.com/think/topics/random-forest

[19] T. C. Nokeri, "Classification Using Decision Trees," *Data Science Revealed*, pp. 139–151, 2021, doi: 10.1007/978-1-4842-6870-4_8.

[20] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, vol. 2018-January, pp. 1–6, Jul. 2017, doi: 10.1109/ICENGTECHNOL.2017.8308186.