EARLY MALWARE DETECTION AND CLASSIFICATION

19CSE495 PROJECT PHASE 2 REVIEW 1 REPORT

Submitted by

Roll number	Name
CB.EN.U4CSE20238	M SATYA CHANDANA SNEHAL
CB.EN.U4CSE20248	RAGHUNANDAN S
CB.EN.U4CSE20264	S ROHIT
CB.EN.U4CSE20269	V NAGOOR

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AMRITA SCHOOL OF COMPUTING

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641112

SEPTEMBER 2023

AMRITA VISHWA VIDYAPEETHAM

TABLE OF CONTENTS

ABSTRACT

Android, being a prime target for malware creators, contends with a daily deluge of new malicious samples seeking to breach security measures in application stores and to infect unsuspecting devices. In response to this escalating challenge, this project delves into the intricacies of Android malware detection and classification, propelled by the imperative to develop automated mechanisms capable of sifting through many suspicious samples. The focus extends beyond mere identification to address the nuanced task of categorizing malware into families. A preliminary exploration of Android malware families forms the foundation, unveiling the critical practices that are essential for constructing effective detection mechanisms.

Motivated by the pressing need to fortify Android devices against evolving threats, the project unfolds with a commitment to significantly contribute to security endeavors. The core purpose is to leverage machine-learning techniques, particularly deep-learning architectures, to construct robust mechanisms for identifying and classifying Android malware. A pivotal aspect of the methodology involves a novel approach of generating images from DEX files, representing entire files and specific data sections. Applying Convolutional Neural Networks (CNNs) to these images yields promising results, showing the potential to develop accurate family classification methods. This study extends its impact by analyzing the resilience of these techniques against adversarial attacks, ensuring their effectiveness in real-world scenarios. In essence, the findings provide a glimpse into the future of Android malware defense, where innovative machine-learning approaches prove instrumental in fortifying devices against the relentless evolution of malicious threats.

1. INTRODUCTION

The escalation of cyberattacks represents a critical and pressing challenge in modern society, tracing its roots back to the emergence of the first virus in the 1970s. In particular, malware has evolved into a potent instrument for large-scale attacks, with a significant impact on users and critical infrastructure. The pervasiveness of these cyber threats is particularly pronounced in the case of smartphones, repositories for vast amounts of private and sensitive data including personal messages, photos, and access to applications such as banking and medical services. Android, with its commanding 80% market share in the global mobile operating system landscape, has emerged as a primary target, bearing 99% of all mobile malware.

The multifaceted nature of Android malware, ranging from scareware to ransomware, has given rise to diverse malware families, compounding this challenge. Addressing this formidable issue requires efficient solutions, and machine-learning and deep-learning techniques for image classification have emerged as powerful tools. These techniques offer a promising avenue for the development of robust malware detectors capable of managing vast application datasets. By leveraging previous training on labelled samples, these detectors can categorize applications as either malicious or benign, thereby providing a crucial layer of defence against the complex landscape of Android malware. The justification for employing such advanced techniques lies in the sheer scale, complexity, and severity of the Android malware problem, which demands sophisticated and scalable solutions to safeguard user data and mobile ecosystems effectively.

Android malware encompasses a diverse range of functionalities, each designed to exploit vulnerabilities and manipulate user behaviour. Some common types of Android malware include:

- Adware: Bombarding users with intrusive advertisements, often leading to unwanted redirects and potential installation of other malicious software.

- Banking malware: Targeting financial information and credentials, allowing attackers to steal money, intercept transactions, and manipulate financial data.
- SMS malware: Exploiting the SMS service to send premium-rate messages, intercept sensitive information received via SMS, and spread further malware through SMS spam.
- Riskware: While not inherently malicious, these programs pose potential security threats due to vulnerabilities or lack of proper security controls, enabling attackers to exploit them for malicious purposes.

With the widespread adoption of Android devices and the vast amount of sensitive data stored within them, effectively detecting and mitigating Android malware has become an imperative task. This malicious software poses a significant threat to user privacy, financial security, and overall device functionality. Robust detection techniques are essential to combat this evolving threat landscape and protect users from the detrimental impacts of Android malware, including:

- **Preserving user privacy:** Detecting and removing malware that steals personal information, contacts, and browsing history helps safeguard user privacy and prevent unauthorized access to sensitive data.
- **Ensuring financial security:** Identifying and neutralizing malware designed to intercept financial transactions, manipulate banking data, and siphon funds protects users from financial losses and fraud.
- **Maintaining device integrity:** Detecting and removing malware that disrupts system operations, drains battery life, and installs additional malware ensures the smooth functioning and optimal performance of Android devices.
- **Preventing further infection:** By identifying and isolating infected devices, effective detection techniques can limit the spread of malware and protect other devices from becoming compromised.

Android malware detection has crucial applications across various sectors, including:

- **Education and Research**: Protecting sensitive student data, intellectual property, and online learning platforms from cyberattacks.
- **Healthcare**: Safeguarding patient records, medical research data, and hospital operations from ransomware and data breaches.
- **Finance/Banking**: Preventing financial fraud, securing customer data, and ensuring the integrity of banking systems against cyberattacks.
- **Government/Military**: Protecting national security information, confidential government documents, and critical military infrastructure from cyber espionage and sabotage.

By implementing effective Android malware detection techniques, these sectors can prevent significant financial losses, reputational damage, and potential harm to individuals and national security.

Android malware detection is a crucial area of research with significant implications for user security and privacy. However, several challenges hinder the development of effective and robust detection techniques.

- **Zero-day Attacks**: New malware strains emerge before researchers can develop signatures or models to detect them, leaving users vulnerable during this critical window.
- **Dynamic Behavior**: Malware can modify its behavior based on the environment and user interaction. This dynamic nature complicates the detection process, as static analysis methods may not capture all malicious activities.
- **Lack of Explainability**: Many machine learning models are complex and lack interpretability, making it difficult to understand their decision-making process. This hinders trust and accountability, as researchers and users may not be able to assess model fairness and potential biases.

This study introduces a novel system that leverages eXplainable Artificial Intelligence (XAI) techniques to address the challenges of zero-day malware detection, dynamic behaviour, and lack of explainability. By combining static analysis with advanced shape modelling and dynamic analysis with a Convolutional Neural Network (CNN), the system achieves improved detection accuracy and enhanced adaptability to evolving threats. Additionally, the use of XAI techniques like SHAP provides increased transparency into the system's decision-making process, allowing researchers and users to understand how it identifies and classifies malware. This combination of features makes the proposed system a valuable tool for combating contemporary cyber threats.

Key Advantages:

- **Improved Detection Accuracy**: Advanced shape modelling techniques lead to a nuanced understanding of malware characteristics, enhancing identification accuracy.
- **Enhanced Adaptability**: CNN-based dynamic analysis allows the system to model and analyse malware behaviour in real-time, adapting to evolving threats and dynamic behaviour.
- **Increased Transparency**: XAI techniques like SHAP provide explanations for classifications, enabling researchers and users to understand the decision-making process and identify potential biases.

2. Literature Survey

2.1 DL-AMDet: Deep Learning-Based Malware Detector for Android

The paper introduces DL-AMDet, a deep learning-based malware detector designed specifically for the Android platform. Leveraging the power of deep learning algorithms, DL-AMDet aims to accurately detect and classify Android malware samples. By analyzing various features extracted from Android applications, including permissions, API calls, and manifest file information, DL-AMDet constructs a comprehensive representation of each application. This representation is then fed into a deep neural network architecture, which learns to distinguish between benign and malicious samples. Experimental evaluations conducted on a diverse dataset of Android applications demonstrate the effectiveness of DL-AMDet in detecting malware with high accuracy. The model's performance surpasses that of traditional machine learning-based detectors, showcasing the advantages of deep learning approaches in malware detection. Furthermore, DL-AMDet exhibits robustness against evasion techniques commonly employed by malware authors to evade detection. In conclusion, DL-AMDet presents a promising solution for combating Android malware threats using deep learning techniques. Its ability to accurately detect malicious applications, coupled with its resilience against evasion tactics, makes it a valuable tool for enhancing the security of Android devices and safeguarding users' privacy and data.

2.2 An Android Malware Detection Method Using Deep Learning based on Multi-features

This study presents an innovative approach to Android malware detection by employing a Bidirectional Long Short-Term Memory (BiLSTM) network, which combines text classification techniques with deep learning methodologies. Authored by an undisclosed team, the method focuses on extracting crucial features

from Android applications to effectively identify malware instances. By analyzing permissions, services, receivers, and intents extracted from the applications, the proposed method achieves an impressive accuracy rate of 97.47%. The study highlights the importance of addressing the ongoing challenges posed by malware targeting the Android operating system, attributed to its widespread usage and inherent openness. Through the integration of text processing techniques and deep learning algorithms, the proposed method automatically extracts features from APKs' static analysis reports, facilitating efficient malware detection. Moreover, the BiLSTM network utilized in the method outperforms alternative techniques such as DPCNN and Fasttext in terms of detection performance. The accuracy ratio and precision metrics further underscore the effectiveness of the proposed approach in accurately identifying Android malware instances. While the study does not explicitly mention specific limitations, it suggests potential avenues for future research, including the incorporation of static and dynamic features to enhance detection capabilities. Overall, the proposed method offers a promising solution to the ongoing challenge of Android malware detection, leveraging advanced deep learning techniques to bolster mobile security measures.

2.3 DL-Droid: Deep Learning-Based Android Malware Detection Using Real Devices

The paper titled "DL-Droid: Deep learning-based Android malware detection using real devices," authored by Mohammed K. Alzaylaee, Suleiman Y. Yerima, and Sakir Sezer, presents an innovative approach to Android malware detection leveraging deep learning techniques. DL-Droid explores the use of deep learning with stateful input generation to achieve exceptional accuracy in detecting malicious applications. Through experiments conducted on real devices, DL-Droid demonstrates remarkable performance, with detection rates reaching up to 99.6%, surpassing traditional methods and outperforming other deep learning approaches. By combining dynamic and static features, the system achieves outstanding results,

with a detection rate of up to 99.6%. Moreover, DL-Droid investigates dynamic stateful input generation, which enhances detection accuracy, achieving up to 97.8% detection rate solely with dynamic features. The study compares DL-Droid with traditional machine learning classifiers, highlighting the superiority of deep learning-based methods for Android malware detection. Furthermore, DL-Droid discusses the limitations of existing malware detection approaches, emphasizing the need for advanced techniques in light of the failure of Google Play Protect and the inefficiency of third-party stores in detecting harmful applications. Through a comparative study with seven popular machine learning classifiers and experimentation on real devices, DL-Droid establishes its effectiveness in detecting Android malware and underscores the importance of stateful input generation for enhanced accuracy. Overall, DL-Droid presents a significant advancement in Android malware detection, offering a potent solution to combat the evolving threat landscape, while advocating for further research in this critical area of mobile security.

2.4 Android Malware Classification by CNN-LSTM

This paper proposes a pioneering approach to Android malware detection utilizing a Convolutional Neural Network (CNN) combined with Long Short-Term Memory (LSTM) networks. Leveraging deep learning techniques, the method achieves remarkable accuracy, reaching 94% on the CICMalDroid2020 dataset, which comprises over 17,000 Android samples. By harnessing the capabilities of CNNs to extract spatial features and LSTM networks to capture temporal dependencies, the hybrid model demonstrates superior performance with minimal false positives, averaging a 93% accuracy rate. The study underscores the pressing need for advancements in mobile security, particularly in the context of Android's market dominance since 2012, with over 2.5 billion users vulnerable to exploitation by cyber-criminals, exemplified by threats like the TangleBot malware. While traditional methods such as static and dynamic analysis have been employed for Android malware detection, the proposed deep learning approach showcases

significant promise for enhancing detection accuracy. Moreover, the paper highlights the challenges associated with gradient descent instability and acknowledges the CNN-LSTM classifier's current limitations in achieving accuracy beyond 94%. The findings underscore the importance of continued research efforts to mitigate the rapid growth of Android malware and improve mobile security standards.

2.5 Android Malware Detection Using Machine Learning

This paper introduces two innovative approaches to combat Android malware. The first method, presented by authors Ayat Droos, Rama Al-Attar, Awss Al-Mahadeen, and Mohammad Ababneh from Princess Sumaya University for Technology in Amman, Jordan, proposes a novel Android malware detection technique utilizing machine learning. By analyzing features extracted from Android application packages (APKs), including permissions requested, API calls, and manifest file contents, the method constructs a comprehensive representation of each APK, capturing both static and dynamic aspects of the application's behavior. Experimental evaluations demonstrate the effectiveness of the approach, achieving high accuracy in distinguishing between benign and malicious applications and outperforming traditional signature-based detection methods. The second method, outlined in the other paper, introduces an innovative approach to classify Android malware families based on features extracted from Dalvik Executable (DEX) file sections. By leveraging both RGB image and plain text representations derived from the DEX file, the method captures spatial distribution through color and texture features, while also extracting statistical features such as opcode frequency and n-grams from the textual representation. A multiple kernel learning algorithm fuses these features into a comprehensive descriptor, enabling a Support Vector Machine (SVM) classifier to achieve accurate familial classification. Experimental evaluations conducted on the Android Malware Dataset (AMD) demonstrate the effectiveness of the proposed method, achieving an impressive accuracy of 96.7% in classifying malware families and surpassing the performance of existing methods. Additionally, the proposed method offers improved efficiency, reducing

the feature extraction time compared to other approaches. In conclusion, both papers present novel and efficient methods for combating Android malware, offering valuable tools for security researchers and developers in the ongoing fight against evolving threats targeting Android devices.

2.6 Detect Android Malware by Using Deep Learning: Experiment and Evaluation"

In a study by Droos and Al-Mahadeen, the effectiveness of four machine learning classifiers, Random Forest, J48, IBK, and Naive Bayes, was evaluated for Android malware detection. The CICMalDroid 2020 dataset, comprising 11,598 instances and 471 features, was used for training and testing. Feature selection was implemented using the Information Gain attribute Evaluator and CFS Subset Evaluator, reducing the feature set to 27 attributes. To address the imbalanced nature of the dataset, SMOTE (Synthetic Minority Oversampling Technique) was applied. The results revealed that Random Forest outperformed the other classifiers, achieving an accuracy of 98.6051, while J48, IBK, and Naive Bayes exhibited accuracies of 96.9207, 92.6097, and 50.658, respectively.

2.7 Explainable AI for Android Malware Detection: Towards Understanding Why the Models Perform So Well?

The paper explores the factors that contribute to the high performance of machine learning (ML)-based models in Android malware detection. While ML has emerged as a promising approach for identifying malicious apps, concerns have been raised regarding the reliability of these models, particularly under unrealistic experimental settings.The authors employ Explainable AI (XAI) techniques to delve into the decision-making processes of ML-based models, analyzing their learned patterns and identifying the key features that influence their predictions. The paper critically examines the high performance of machine learning (ML)-based models in Android malware detection, addressing concerns about their reliability under unrealistic experimental conditions. By employing Explainable AI (XAI) techniques, the

authors uncover a significant factor contributing to over-optimistic classification performance – temporal sample inconsistency in the training dataset. The study outlines a methodology involving data collection, ML model training, and XAI-based analysis to reveal that models often rely on temporal-related features rather than actual malicious behaviors, leading to inflated accuracy in controlled settings. The research emphasizes the importance of understanding these limitations and advocates for the integration of XAI techniques to enhance the robustness and reliability of ML-based Android malware detection systems. It calls for a shift in focus from high-accuracy metrics to factors like generalizability and interpretability, anticipating that these findings will inspire future research in the cybersecurity domain.

The conclusion is that existing research on ML-based Android malware detection overestimates its performance due to unrealistic experimental setups. This overestimation is caused by the models' reliance on temporal inconsistencies in the training data rather than actual malicious behaviors. The paper highlights the importance of utilizing Explainable AI (XAI) techniques to understand the limitations of these models and develop more robust and reliable detection systems. It encourages the community to move beyond focusing solely on high-accuracy metrics and consider factors like generalizability and interpretability. The paper concludes by expecting that its findings will inspire future researchers to utilize XAI techniques to address underlying issues in ML/DL-based systems.Overall, the paper presents a significant critique of existing research on ML-based Android malware detection and proposes a path for improvement through XAI. This work has the potential to significantly impact the field of cybersecurity.

4. Proposed Methodology

This chapter describes the preliminaries/ existing baseline works followed with the details of dataset used for the development of the proposed methodology and discusses the proposed method with the architectures in detail. In addressing the complex landscape of Android malware detection, our proposed methodology unfolds as a strategic fusion of insights gleaned from a comprehensive review of existing research. This endeavour is grounded in the recognition that a multifaceted approach is imperative for effectively identifying and combating evolving malware threats.

Preliminaries:

- Chen et al. from Guangzhou University developed an Android malware detection method using BiLSTM networks, achieving a high accuracy of 97.47%. Their approach combines text processing with deep learning and utilizes the Androguard tool for feature extraction. Experimental results show superior performance compared to other methods, with future considerations including the integration of static and dynamic features for enhanced detection.

- Nasser, Hasan, and Humaidi present DL-AMDet, a deep learning-based malware detection system for Android, employing CNN-BiLSTM and Autoencoders. Their approach achieves an impressive 99.935% accuracy by combining static and dynamic analysis. DL-AMDet outperforms existing techniques, overcoming static analysis limitations and detecting obfuscation and Zero-Day threats effectively. By leveraging CNN-BiLSTM for static analysis and Deep Autoencoders for dynamic analysis, DL-AMDet offers a

comprehensive solution for Android malware detection, addressing both known and emerging threats.

- Amenova, Turan, and Zharkynbek introduce an Android malware classification method utilizing CNN-LSTM, achieving a notable accuracy of 94%. Their work underscores the significance of deep learning in enhancing Android malware research for market quality improvement. By employing a hybrid CNN-LSTM model on the CICMalDroid2020 dataset, their approach demonstrates effectiveness in predicting Android malware with minimal false positives. Encouraging further exploration, their study advocates for ongoing research to combat malware proliferation and bolster security measures.

- Sahın et. al used multiple linear regression methods to classify android malware based on permission features.The authors applied their method to four datasets, two of them were collected by the authors. Furthermore, they used the ensemble learning/bagging method to create classifiers. The results showed that the highest accuracy was 98.53% using ensemble learning on the second dataset they collected.

- Bai et. al proposed a novel approach that applies N-gram semantic neural method using multi-filter CNN and GRU to detect malicious network traffic generated by mobile malware. The proposed method applied N-gram semantic neural model using a convolutional neural network (CNN) model, and gated recurrent unit (GRU) on a dataset collected from the Canadian Institute of Cybersecurity. Their model achieved an accuracy of 92.6%.

- Fang et. al used a fusion algorithm based on multiple kernel learning to classify android applications. They extracted the features from the Dalvik

Executable (DEX) file section. To do that, the features from DEX file were converted into RGB images and plain text. Then, the features were extracted from the texture and colour of images and text. Their approach was applied on Android Malware Dataset (AMD)[10], and the accuracy achieved was 96%.

- X. Wang and C. Li , proposed a new Android malware detection framework called "KerTSDroid" generating their own dataset of 191,250,000 data records extracted from 12,750 Android applications, 6375 Malicious Application of 38 malware families, and 6375 Benign Application of famous 24 Android Market application category. One hundred and twelve kernel features were used and analyzed through "task_struct" after that data preprocessing and cleaning were implemented, where then data was translated into numerical values to normalize the data collected from Android application. The tests were implemented in Apache Hadoop v2.6.0 and Apache Spark v1.6.0. Various Machine-Learning classifiers were used including: Naïve Bayes (NB), Logistic Regression (LR), Decision Tree (DT), and Support Vector Machine (SVM).

- Zhu et al. proposed a malware detection model using stacking ensemble classifiers. The model proposed starts by extracting features using static analysis. The training dataset of the model is divided into subsets, where each subset then undergoes principal component analysis to ensure individual diversity. The stacking ensemble learner is a two layered architecture. The first tier uses MLP classifiers as the base learners while the second layer uses SVM that is used for the final prediction result. The authors have tested the performance of the model using two datasets. The first dataset was a real dataset with multi-level features where an accuracy of 87.89% was achieved while the second dataset was based on MUDFLOW and the model achieved an accuracy of 97.04%.

Models Implemented and Its Analysis

XGBoost:

XGBoost, an acronym for eXtreme Gradient Boosting, stands out as a powerful machine learning algorithm renowned for its efficiency in processing and analyzing large datasets. Its notable strength lies in its adeptness at handling high-dimensional feature spaces, making it particularly well-suited for tasks that involve extensive feature extraction. In the context of the CICMaldroid dataset, which encompasses a diverse array of features such as permissions, API calls, file system activity, and code characteristics, XGBoost's capability to efficiently manage complex feature relationships becomes paramount. The computational efficiency of XGBoost is a significant advantage, ensuring that feature extraction can be executed with minimal computational overhead, a crucial consideration for large and intricate datasets.

Working Mechanism of XGBoost:

XGBoost operates based on an ensemble learning approach, combining the strengths of multiple weak learners to create a robust and accurate predictive model. It sequentially builds a series of decision trees, each compensating for the errors of its predecessors. The process involves assigning weights to misclassified instances, emphasizing the importance of correcting mistakes. Notably, XGBoost introduces regularization terms in its objective function, controlling the complexity of the model and preventing overfitting. Its key innovation lies in the integration of gradient boosting techniques, optimizing the model's performance through the iterative refinement of weak learners. This iterative process, coupled with parallel computing capabilities, contributes to the algorithm's efficiency, making XGBoost a preferred choice in various machine learning applications, including feature-rich datasets like CICMaldroid.

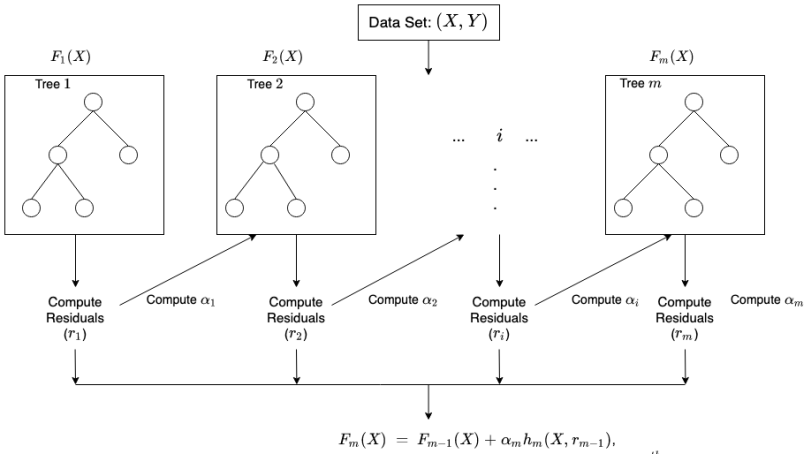

Figure 1: XGBoost Architecture

Random Forest:

Random Forest stands as a formidable ensemble learning algorithm widely recognized for its efficacy in various machine learning applications. In the specific context of the CICMaldroid dataset, Random Forest emerges as a powerful tool for feature extraction, showcasing several advantages that make it a preferred choice. Random Forest excels in handling the intricacies of large datasets, and its adaptability to diverse feature sets positions it as an ideal candidate for feature extraction within the CICMaldroid dataset. The algorithm's robustness in managing high-dimensional feature spaces is particularly noteworthy, making it well-suited for the complex task at hand.

Working Mechanism of Random Forest:

Random Forest operates on the principle of ensemble learning, wherein it combines the outputs of multiple decision trees to enhance predictive accuracy and robustness. In the feature extraction phase, Random Forest assesses the importance of each feature, assigning weights based on their contribution to the predictive performance. This ability to discern feature importance is pivotal for extracting

meaningful insights from the CICMaldroid dataset, which includes a diverse array of features such as permissions, API calls, file system activity, and code characteristics.

The ensemble nature of Random Forest contributes to its resilience against overfitting and noise, a critical advantage in the feature extraction process. By aggregating the predictions of multiple trees, Random Forest minimizes the risk of individual trees capturing noise or specific patterns, ensuring the extracted features are more robust and generalizable. Thus, Random Forest emerges as a versatile and powerful algorithm for feature extraction, especially in scenarios like the CICMaldroid dataset with intricate feature relationships.

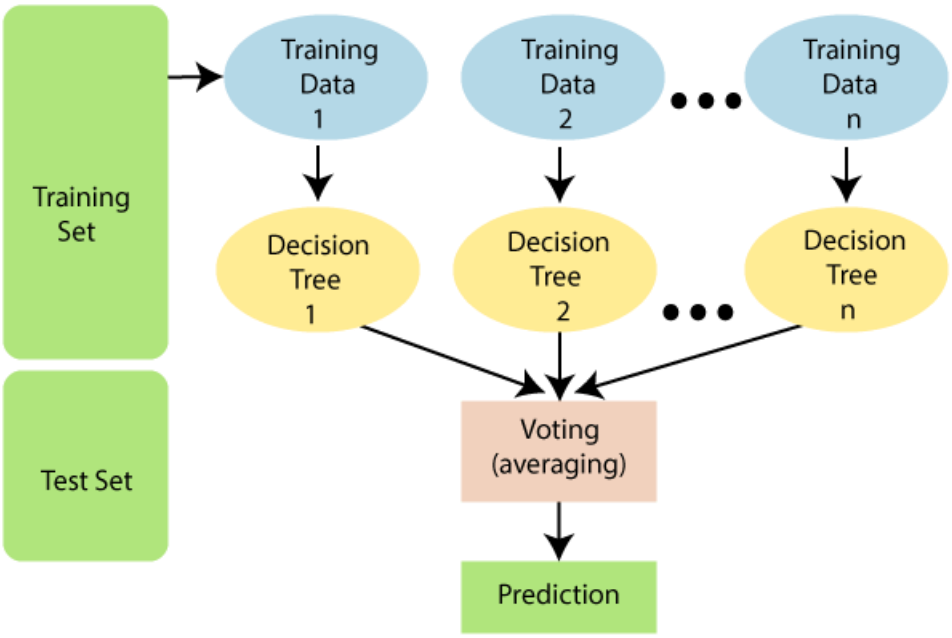

Figure 2: Random forest Architecture

LightGBM:

Light Gradient Boosting Machine (LightGBM) stands out as a high-performance gradient boosting framework, specifically designed for efficiency and scalability. In the domain of feature extraction, particularly within the complex landscape of datasets like CICMaldroid, LightGBM emerges as a potent tool with distinct advantages.

LightGBM is tailored for optimal performance in feature extraction, excelling in scenarios involving large and intricate datasets such as CICMaldroid. Its proficiency in handling high-dimensional feature spaces makes it an excellent choice for extracting essential features efficiently.

Working Mechanism of LightGBM:

LightGBM employs a histogram-based learning approach and a unique leaf-wise tree growth strategy, setting it apart in terms of speed and accuracy. The histogram-based learning optimizes the binning process, contributing to faster computation of information gain during the tree-building process. The leaf-wise growth strategy prioritizes nodes with higher information gain, further enhancing the overall efficiency of the algorithm.

LightGBM's scalability is a notable feature, enabling feature extraction without imposing significant computational overhead. Its ability to manage complex feature relationships is complemented by its provision of interpretable feature importance scores. This unique combination of speed, scalability, and interpretability positions LightGBM as an asset in the feature extraction process, particularly when dealing with intricate datasets like CICMaldroid.

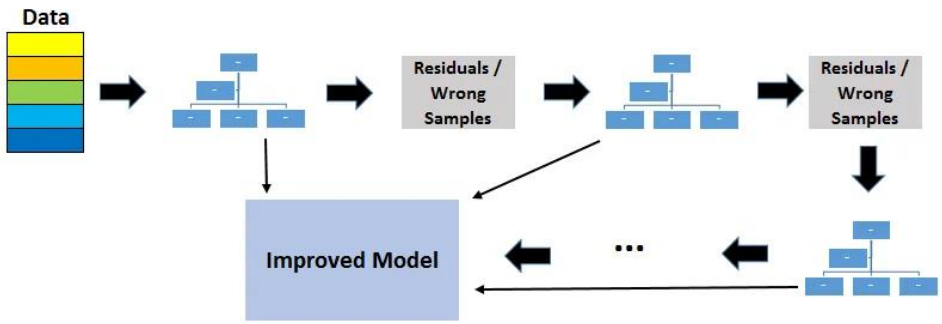

Figure 3: Light Gradient Model

For Deep Learning Classifiers:

BiLSTM (Bidirectional Long Short-Term Memory):

BiLSTM (Bidirectional Long Short-Term Memory) stands out as a robust deep learning architecture, particularly effective in the domain of malware detection, leveraging the intricacies of the CICMaldroid dataset. Similar to Random Forest, BiLSTM demonstrates remarkable capabilities in extracting features crucial for distinguishing malware instances within the dataset.

Working Mechanism of BiLSTM:

BiLSTM operates on the principles of recurrent neural networks (RNNs), augmented with memory cells that facilitate the retention of sequential information over extended periods. In the feature extraction phase, BiLSTM dynamically learns the temporal dependencies within the dataset, crucial for understanding the behavioral patterns indicative of malware presence. The bidirectional nature of BiLSTM enables it to capture contextual information from both past and future data

points, enhancing its ability to discern subtle but significant patterns embedded within the CICMaldroid dataset.

Architechture of BiLSTM:

The architecture of BiLSTM comprises multiple LSTM layers, each equipped with memory cells capable of retaining and selectively passing information. Through iterative forward and backward passes, the model refines its understanding of sequential data, progressively extracting higher-level features relevant to malware detection. Additionally, dense layers integrated into the architecture enable the transformation of extracted features into representations conducive to classification.

Model Architecture:

Input Layer:

- Shape: (number of samples, number of timesteps, number of features)
- This layer accepts sequences of feature vectors representing malware samples.

Bidirectional LSTM Layers:

- Bidirectional LSTM layer with tunable units (number of neurons) and return_sequences=True to return sequences for subsequent layers.
- Multiple Bidirectional LSTM layers are stacked to capture temporal dependencies in both forward and backward directions.

Dense Layers:

- Dense layer with tunable units and activation function.
- Dropout layer to mitigate overfitting by randomly dropping a fraction of input units.
- Another Dense layer with tunable units and activation function.

Output Layer:

- Dense layer with softmax activation function.

- Output units equal to the number of classes (assuming a multiclass classification problem).
- Softmax activation ensures the output represents class probabilities.

Training Procedure:

- Hyperparameter Tuning: Hyperparameters are tuned using RandomSearch from the Keras Tuner library to optimize validation accuracy.
- Validation Split: 20% of the training data is used for validation during training.
- Epochs: The model is trained over 100 epochs to update parameters and optimize performance.
- Batch Size: Training is performed in mini-batches of size 512 for efficient computation.

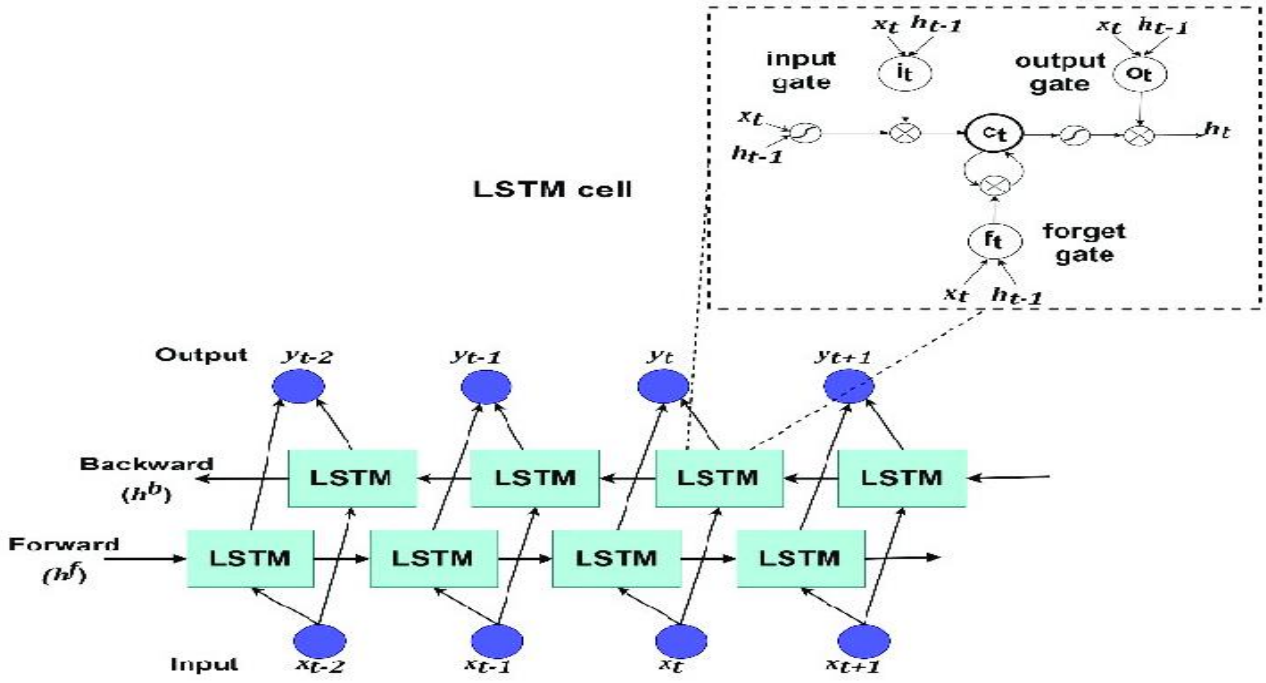

Figure 4: BiLSTM Model

HYPERPARAMETER TUNING RESULTS:

Best hyperparameters:

- 'units_lstm_1': 224
- 'units_lstm_2': 32
- 'units_lstm_3': 32
- 'units_dense_1': 96
- 'activation_dense_1': 'tanh'
- 'dropout_1': 0.4
- 'units_dense_2': 32
- 'activation_dense_2': 'relu'
- 'optimizer': 'adam'
- 'learning_rate': 0.0018054519740046478

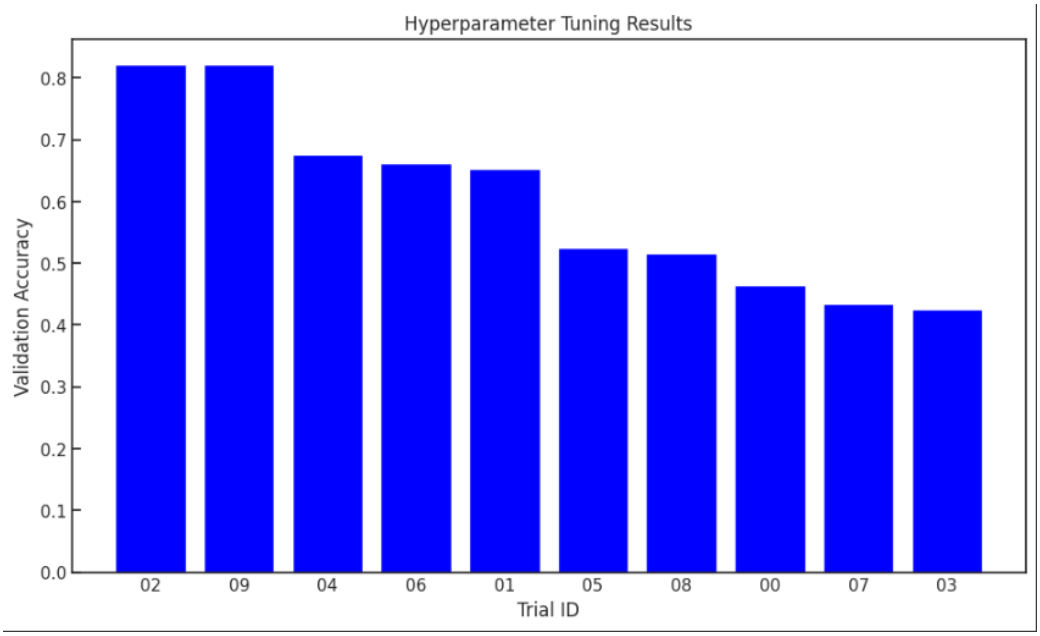

Figure 5: Hyper parameter tuning result of BiLSTM Model

LSTM-CNN (Long Short-Term Memory - Convolutional Neural Network):

LSTM-CNN represents a fusion of two powerful deep learning architectures, combining the sequential processing capabilities of LSTM with the feature extraction prowess of CNN. Within the context of malware detection using the CICMaldroid dataset, LSTM-CNN emerges as a sophisticated model capable of effectively extracting discriminative features indicative of malware behavior. The system takes advantage of NLP methods as a baseline, mixes CNNs and LSTM neurons to capture local spatial correlations, and learns from successive long-term dependencies

Mechanism of LSTM-CNN:

LSTM-CNN operates by first processing the sequential nature of the data through LSTM layers, enabling the model to capture temporal dependencies inherent in the dataset. By retaining memory over extended sequences, LSTM layers provide a comprehensive understanding of the sequential patterns characteristic of malware activities within the CICMaldroid dataset. Subsequently, the extracted features undergo further refinement through convolutional layers, which excel in identifying spatial patterns across the feature space.

MODEL ARCHITECHTURE:

Input Layer:

- The input layer accepts sequences of feature vectors representing malware samples. Each feature vector is represented as a single timestep.
- Shape: (number of samples, number of timesteps, 1)

LSTM Layer:

- The LSTM (Long Short-Term Memory) layer processes the sequential input data and returns sequences for subsequent layers.
- Number of units: Tunable between 32 to 256 (Random Search)
- Return Sequences: True that is it will output all the hidden states of each time steps.

Convolutional Layer:

- The Conv1D layer applies convolutional operations to the output sequences from the LSTM layer to extract spatial features.
- Number of filters: Tunable between 16 to 64 (Random Search)
- Kernel Size: Tunable between 3 to 5 (Random Search)
- Activation: ReLU

Flatten Layer:

- The Flatten layer flattens the output from the convolutional layer that is used to make the multidimensional input one-dimensional to prepare it for the dense layers.

Output Layer:

- The output layer produces the final classification output, with each unit representing the probability of a particular class.
- Units: 5 (Assuming it's a multi-class classification problem)
- Activation: Sigmoid

Training Process:

The input data, both training, and testing sets are reshaped to match the input requirements of the LSTM-CNN architecture. The model is compiled with the specified optimizer (Adam or RMSprop), categorical cross-entropy loss function, and accuracy metric. The model is trained using the training data. Hyperparameters are tuned via Random Search, optimizing for validation accuracy. Training is performed for a specified number of epochs (100) to iteratively update the network's

parameters and optimize performance and with a defined batch size (512) to efficiently process data in mini-batches. After training, the model's performance is evaluated using the test data. Various classification metrics such as accuracy, precision, recall, F1 score, specificity, ROC AUC score and MCC are computed. A classification report and confusion matrix are generated to assess the model's performance visually.This comprehensive training process and model architecture allow the LSTM-CNN model to effectively capture both sequential and spatial features from the input data, making it suitable for malware detection and classification tasks.

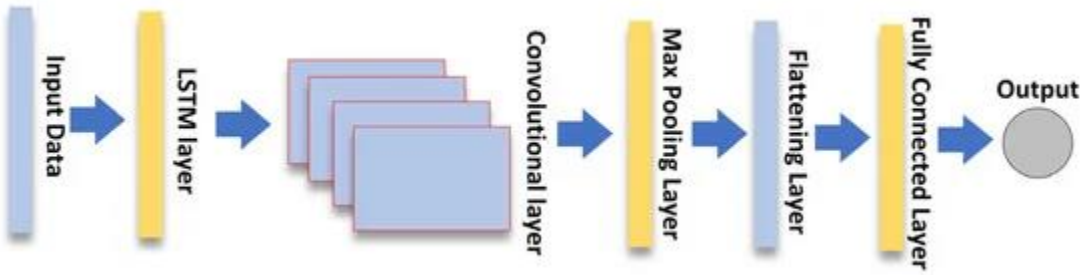

Figure 6: LSTM-CNN Model

Hyperparameter Tuning:

To optimize the performance of LSTM-CNN, hyperparameter tuning plays a crucial role. Through techniques such as random search, the model's architecture and optimization parameters are fine-tuned, leading to improved classification accuracy and generalization capabilities. By exploring the hyperparameter space, LSTM-CNN can adapt to the unique characteristics of the CICMaldroid dataset, further enhancing its efficacy in malware detection tasks.

Best Hyperparameters:

- 'units_lstm': 224
- 'filters_conv1d': 16
- 'kernel_size_conv1d': 3
- 'optimizer': 'adam'
- 'learning_rate': 0.008847108230417601

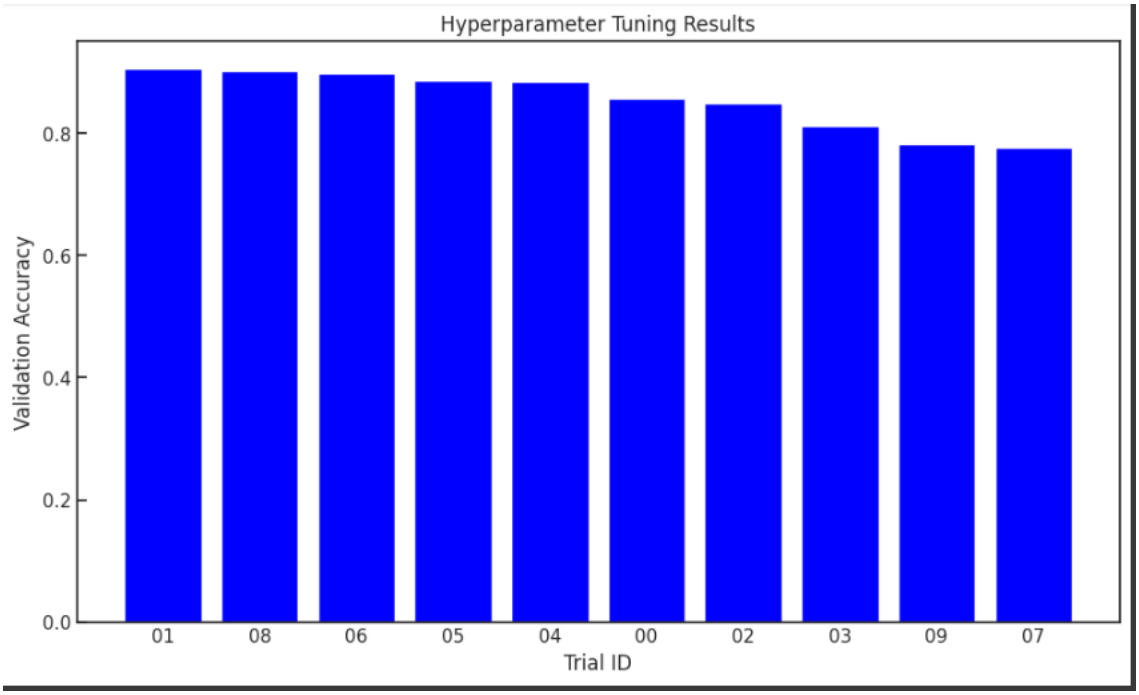

Figure 7: Hyper parameter tuning result of LSTM-CNN Model

Bi-LSTM-CNN (Bidirectional Long Short-Term Memory - Convolutional Neural Network):

Bi-LSTM-CNN is a sophisticated deep learning architecture that integrates bidirectional LSTM layers with convolutional neural network (CNN) layers. In the context of malware detection using the CICMaldroid dataset, Bi-LSTM-CNN offers a powerful solution for capturing both temporal and spatial patterns present in the data.

Mechanism of Bi-LSTM-CNN:

Bi-LSTM-CNN begins by processing the sequential nature of the data through bidirectional LSTM layers. These layers enable the model to capture not only the forward but also the backward temporal dependencies in the input sequence, providing a comprehensive understanding of the sequential patterns inherent in malware behavior. Subsequently, the features extracted by the LSTM layers are further refined through convolutional layers, which excel in identifying spatial patterns across the feature space. By combining these two architectures, Bi-LSTM-CNN can effectively capture both long-term dependencies and spatial relationships within the input data.

Model Architechture:

Input Layer:

- Shape: (number of samples, number of timesteps, 1)
- Accepts sequences of feature vectors representing malware samples. Each feature vector corresponds to a single timestep.

Bi-LSTM Layers:

First Bi-LSTM Layer:

- Number of units: 224

- Return Sequences: True
- Processes the sequential input data bidirectionally and returns sequences for subsequent layers.

Second Bi-LSTM Layer:

- Number of units: 32
- Return Sequences: True
- Further processes the sequential data bidirectionally.

Convolutional Layer:

- Filters: 32
- Kernel Size: 3
- Applies convolutional operations to the output sequences from the Bi-LSTM layers to capture spatial features.

Global Max Pooling Layer:

- Performs global max pooling over the temporal dimension to reduce the spatial dimensions of the feature maps.

Dense Layers:

First Dense Layer:

- Units: 96
- Activation: Tanh
- Introduces non-linearity to the model.
- Helps in learning complex patterns in the data.

Dropout Layer:

- Rate: 0.4
- Regularization technique to prevent overfitting by randomly dropping a fraction of input units during training.

Second Dense Layer:

- Units: 32
- Activation: ReLU
- Another layer of non-linearity to the model.

Output Layer:

- Units: 5
- Activation: Softmax
- Produces the final classification output, with each unit representing the probability of a particular class.

Training process:

The model is compiled using the Adam optimizer with a learning rate of 0.00180 and categorical cross-entropy loss function.Training is performed for 100 epochs with a batch size of 128.The training data is passed through the model to update its parameters iteratively.Validation data is used to monitor the model's performance during training and prevent overfitting.The trained model is evaluated using the test data to compute various classification metrics, including accuracy, precision, recall, F1 score, specificity, ROC AUC score and Matthews Correlation Coefficient.ROC curves are plotted for each class to visualize the model's performance in multi-class classification.The architecture leverages both sequential information (captured by Bi-LSTM layers) and spatial information (captured by the Convolutional layer)from the input data, enabling effective malware detection and classification.

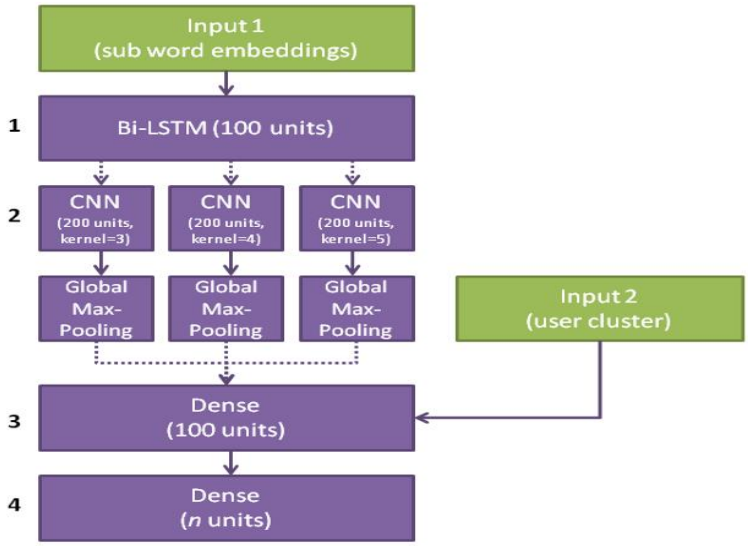

Figure 8: BiLSTM-CNN Model

Hyperparameter Tuning:

Hyperparameter tuning plays a crucial role in optimizing the performance of Bi-LSTM-CNN. Techniques such as random search allow for the exploration of the hyperparameter space, leading to fine-tuned architectures and optimization parameters tailored to the unique characteristics of the CICMaldroid dataset. By adapting to the dataset's complexities, Bi-LSTM-CNN can achieve improved classification accuracy and generalization capabilities.

Best hyperparameters:

- 'units_lstm': 256
- 'filters_conv1d': 256
- 'kernel_size_conv1d': 3
- 'units_dense_1': 256
- 'units_dense_2': 192
- 'learning_rate': 0.0028481438584529698

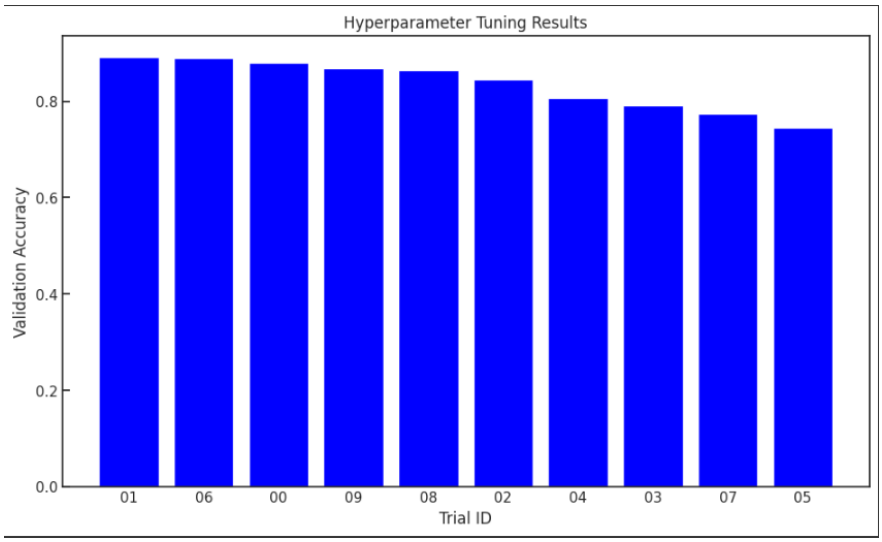

Figure 8: Hyper parameter tuning result of BiLSTM-CNN Model

Explainable AI:

In the pursuit of enhancing the transparency and interpretability of our Android malware detection model, Explainable AI (XAI) plays a pivotal role. By integrating SHAP (SHapley Additive exPlanations) techniques, we aim to provide clear and comprehensive insights into individual predictions, contributing to a more understandable and trustworthy model.

Operational Mechanism of SHAP in Android Malware Detection:

The integration of SHAP in the context of early Android malware detection brings about a profound shift in the interpretability of machine learning models. SHAP operates based on Shapley values, a concept derived from cooperative game theory. In the realm of machine learning, Shapley values assign a unique contribution to each feature for a particular prediction, considering all possible combinations. For each prediction made by the Android malware detection model, SHAP computes the Shapley values for every feature, revealing the individual impact of each feature on the prediction outcome. These values are then aggregated to form comprehensive explanations for the model's decision. The positive or negative contribution of each

feature is quantified, providing a nuanced understanding of why a specific prediction was generated.

The explanations generated by SHAP shed light on the intricate interplay between various features and their influence on the model's output. This approach empowers users, analysts, and developers to gain insights into the decision-making process, fostering a deeper understanding of the model's reasoning. The transparency offered by SHAP not only enhances trust in the Android malware detection system but also facilitates proactive measures and improvements based on a clear understanding of model behavior. In essence, the incorporation of SHAP in Android malware detection represents a crucial step towards achieving Explainable AI, ensuring that the model's decisions are not only accurate but also comprehensible and justifiable.

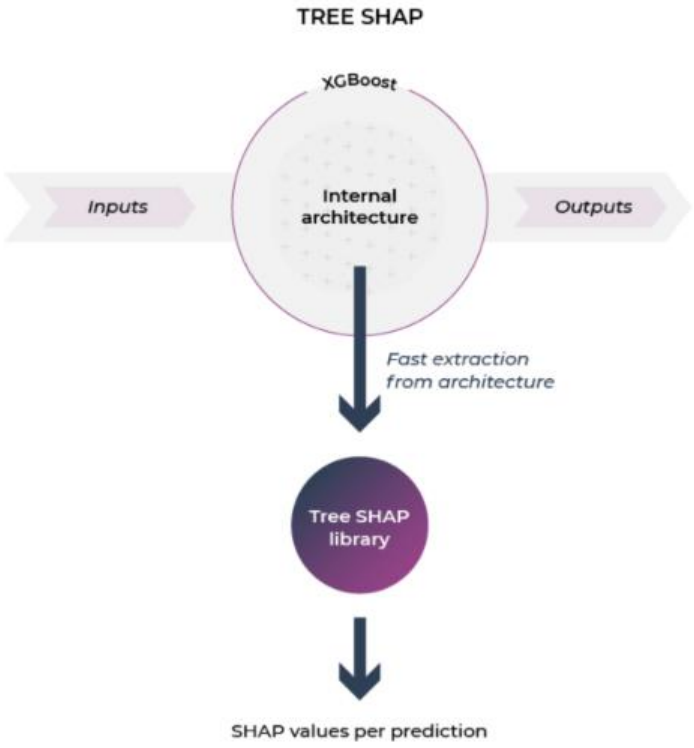

Figure 9: Shap Architecture

The workflow is pictorially represented in the Figure below

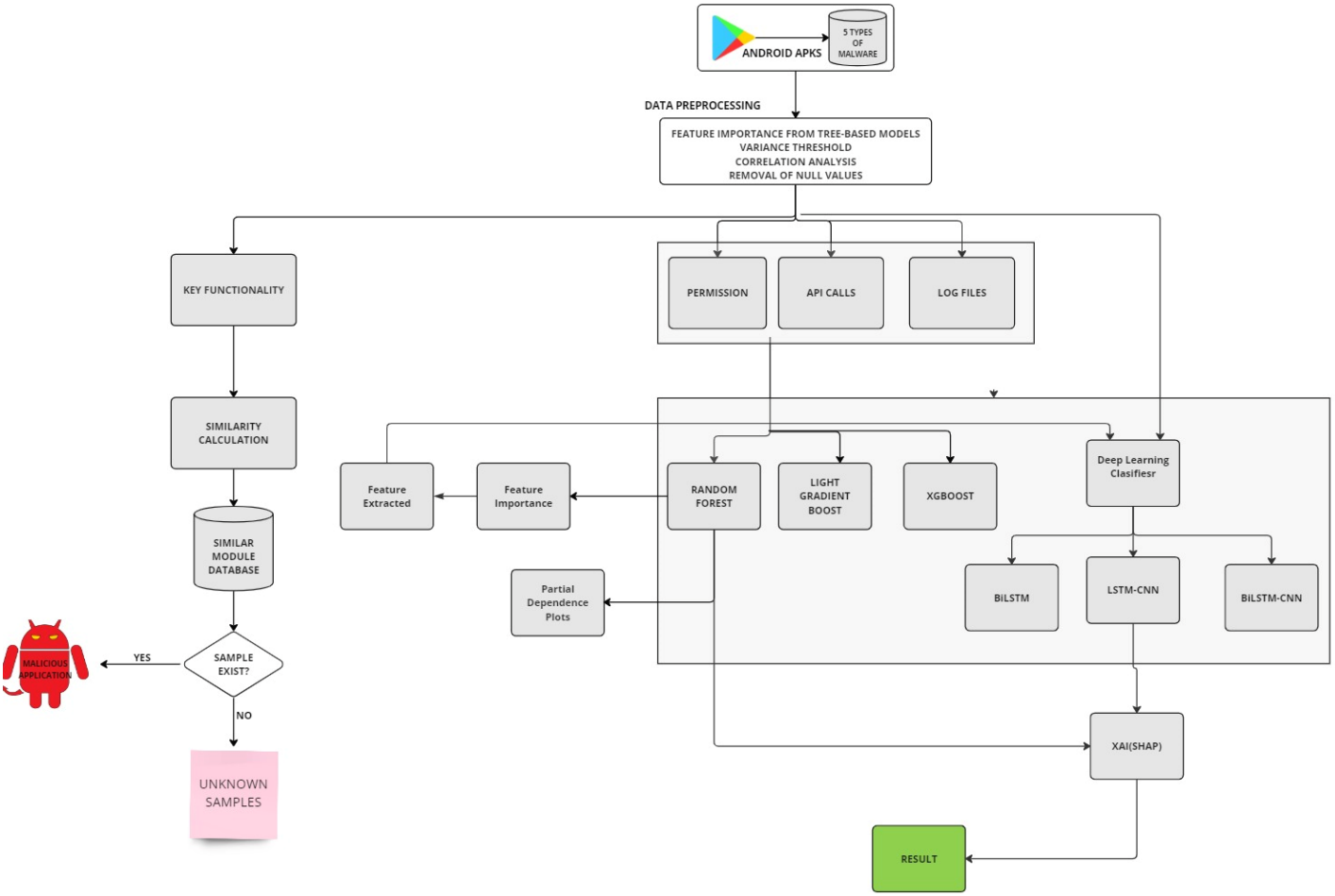

Figure 10: The workflow of proposed method / The proposed workflow Architecture diagram

Dataset Description:

The Android Malware dataset, CICMalDroid 2020, comprises over 17,341 recent and sophisticated samples until 2018. The dataset is characterized by its diversity, spanning five categories: Adware, Banking malware, SMS malware, Riskware, and Benign. Notably, it offers a comprehensive collection of static and dynamic features, encompassing 470 extracted features for 11,598 APK files. This feature set captures frequencies of system calls, binders, and composite behaviors, providing a rich source for training and evaluating machine learning models.

The proposed methodology follows a systematic workflow to achieve early malware detection. We commence with thorough data preparation, addressing quality and imbalance issues in the CICMaldroid dataset. Feature extraction involves the strategic use of Random Forest analysis to identify crucial features, optimizing the input space for subsequent models. Rigorous validation through cross-validation and real-world testing ensures the robustness of the hybrid model. Finally, Explainable AI techniques are integrated to enhance interpretability and elucidate the decision-making process within the hybrid model. This work, titled "Early Malware Detection and Classification," seeks to advance the state-of-the-art in Android malware detection through a synergistic integration of proven methodologies and innovative techniques.

Developing a hybrid model for robust malware detection in the CICMaldroid dataset by integrating feature extraction and deep learning approaches.

Model Implementation:

1) Data Pre-processing:

In this phase the imported datasets were checked for the nan, null values and the handling of any such values were taken into consideration. The target class that the classification of the application was encoded using label encoding where value 1 was assigned for the malicious application adware and 2 value was assigned for banking application and so on. The dataset was then checked for skewness of the data and the skewness transformation was handled using cube root method. The datasets used in this research were balanced datasets that means the samples for the malicious applications were almost equal as compared to the benign application which would have otherwise impacted the efficiency and accuracy of the trained model.

2) Feature Importance:

Feature importance is method in which the input features which are highly responsible for predicting the dependent variable is calculated by assigning scores to the input features. The feature important method is widely used to get insights of the model, data. To reduce the dimensionality and improve the model efficiency the feature importance scores are used. In this research,the Random forest algorithm was used to calculate the features which were responsible for predicting the target values i.e whether the application is malicious or non malicious. The Random forest algorithm was applied on both static and dynamic models where top 50 features were selected for training. To enhance the model's efficiency, features with low variance were pruned, and those exhibiting high correlation within each other were carefully eliminated. This meticulous curation of features aimed to streamline the input space for the subsequent machine learning models.

3) Data Splitting:

For the models the dataset was split into two parts i.e., train data and test data into 80- 20% ratio. where for model training the LSTM-CNN and BiLSTM-CNN model 80% split data was used and after model was trained the testing of the model was carried out using 20% of the data.

4) Model Training:

After preprocessing the data and identifying the 50 most important features using Random forest, the model training phase began. The dynamic model was trained utilizing these important features, incorporating a recurrent neural network (RNN) approach, particularly leveraging Long Short-Term Memory (LSTM) techniques. In the modeling phase, a two-layer LSTM architecture was defined, with each layer comprising 32 units. The Rectified Linear Unit (ReLU) activation function was applied in both LSTM layers. To facilitate information flow between layers and prevent overfitting, the return sequence parameter was set to True in the first LSTM layer, allowing its output to serve as input for the subsequent layer. Additionally, dropout regularization, excluding 20% of data for input connections during weight updates, was implemented after defining each LSTM layer. Following the LSTM layers, a convolutional neural network (CNN) layer was introduced, featuring a convolutional layer with 16 filters and a kernel size of 3, activated by ReLU. This was followed by a flatten layer to prepare the data for the subsequent dense layers. The output layer was defined with a single unit and a sigmoid activation function, suitable for binary classification tasks. The model was compiled using the Adam optimizer and categorical crossentropy loss function. Finally, the model was trained on the training data for 100 epochs with a batch size of 256, while monitoring performance on the validation data.

5) Evaluation:

The following section elaborates the model efficiency and final outcomes of the models. The efficiency was calculated using confusion matrix, the accuracy and loss were graphically plotted after validation of the models. The sections overall cover the results of the research followed in this project. In deep learning, confusion matrix is often used to conclude the performance of the model. In this research the confusion matrix was calculated for dynamic model along with the ROC & AUC scores.

6) Explainable AI Integration:

Incorporate Explainable AI techniques to elucidate the decision-making process, fostering a deeper understanding of the hybrid model's rationale and enhancing interpretability.

5. Data Science interpretation for Malware detection

This part explores the application of data science techniques for interpreting and understanding a malware dataset, specifically the CICMaldroid 2020 dataset. This dataset consists of 471 features and aims to classify mobile applications into five categories: Adware, Banking, SMS Malware, Riskware, and Benign. The primary objective is to investigate how individual features within the dataset influence the model's classification of these app categories.

To explore how specific variables in the dataset affect the model's classification of different app categories, Random Forest Classification is a useful tool. Each tree in the random forest tracks how often a particular feature is used to split data at nodes. This process essentially measures a feature's **contribution to improving the purity** (reducing impurity) of the data at each split.

Feature importance score is based on feature importance of those nodes where the split happened due to a feature and then divide it by the feature importance of all the nodes. To calculate feature importance using Random Forest we just take an average of all the feature importances from each tree.

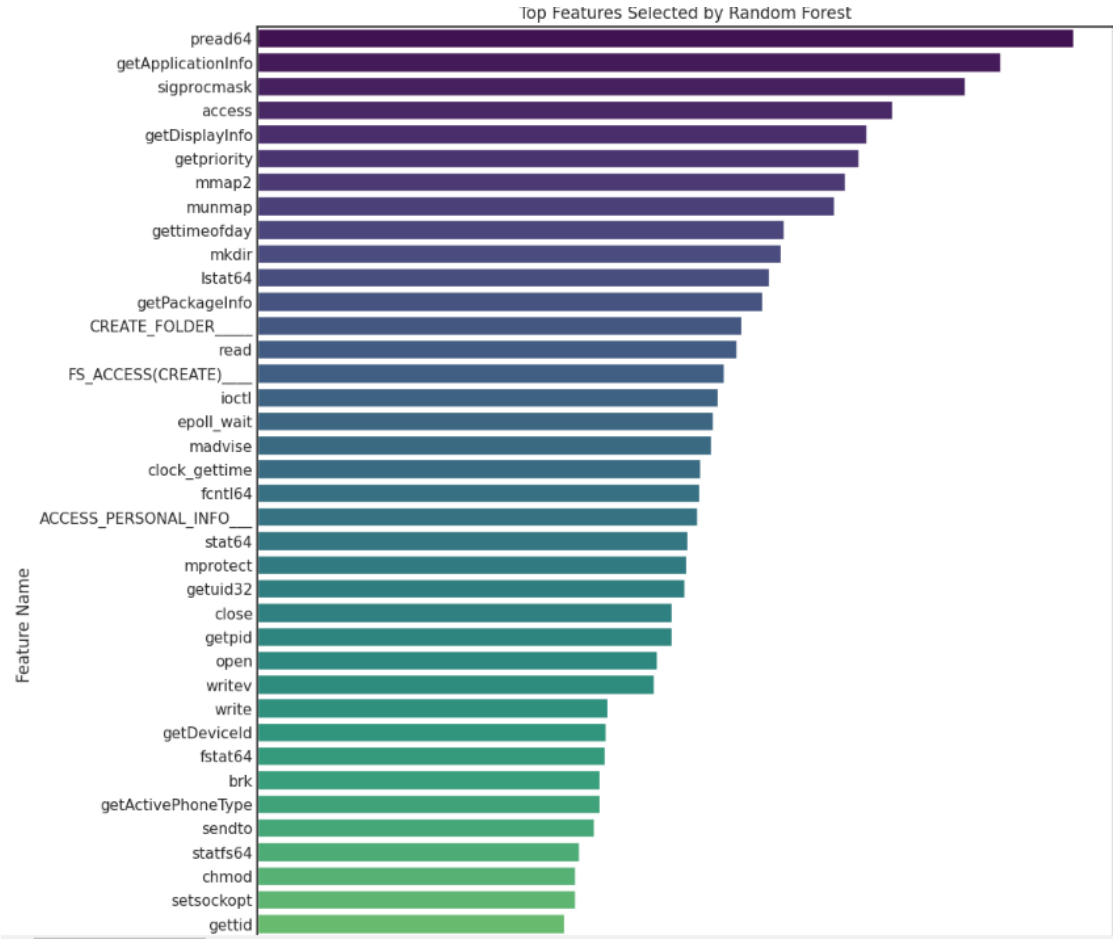

Figure 11: Feature Importance Score barplot

Partial Dependence Plots (PDPs) are visual tools used to explore the relationship between a specific feature and the predicted outcome in a **machine learning model**,

specifically in **classification** and **regression** problems. In multi-class problems, PDPs are created for **each class** to understand how the feature influences the probability of predictions belonging to that class. The plot shows the **average change** in the predicted outcome (i.e., probability for classification) as the value of the specific feature changes along the x-axis. The y-axis displays the corresponding predicted class. While PDPs can be used with various models, **random forests are particularly well-suited** for them due to their inherent feature importance measures.

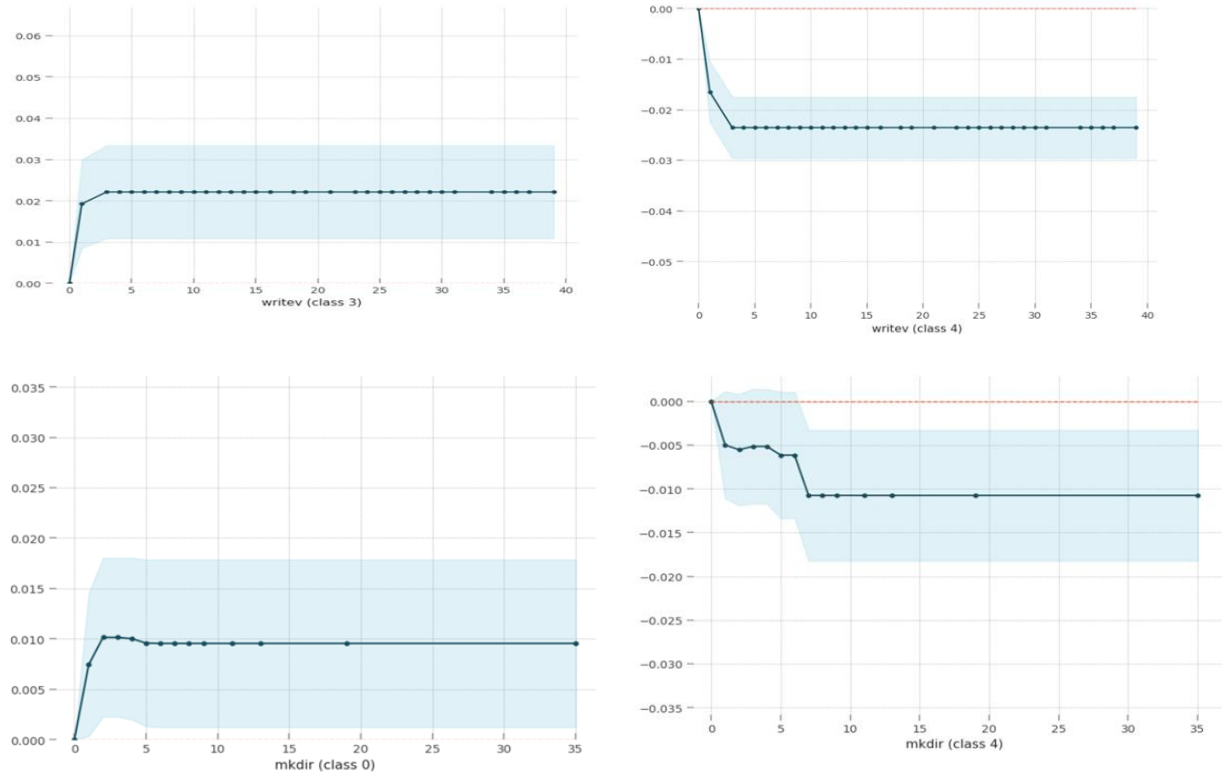

Figure 12: Sample Partial Dependency Plots

Class	Increasing Chance	Decreases Chance
Adware	getpriority, mkdir, writev, ioctl	access
Banking	access, getline1number	gettimeofday
SMS Malware	getline1number	
Riskware	getDisplayinfo, munmap, mkdir, writev, getDeviceId	
Benign	access, getPackageInfo	getpriority, mkdir, getline1number, writev, ioctl

Table 1: PDP report

6. Results and Discussion

This chapter demonstrates the results of the Outcomes of Android Malware Detection through Deep learning models

The purpose of this study was to develop and evaluate a deep learning model for the detection of Android malware, aiming to enhance the security measures in mobile devices.

Four key metrics are utilized to evaluate the performance of the machine learning models in this study: Accuracy, Recall, and Precision and F1-Score These metrics provide insight into various aspects of the models' performance and assist in identifying the most effective model for the task of Android malware detection.

Accuracy: It represents the overall proportion of correct predictions made by the model. Mathematically, it can be expressed as:

$$Accuracy = (TP + TN)/(TP + TN + FP + FN)$$

A higher accuracy value indicates that the model is more adept at correctly identifying both benign and malicious applications.

Recall: This metric focuses on the model's ability to identify all actual positive cases (malicious applications). It is calculated as:

$$Recall = TP/(TP + FN)$$

A high recall value indicates that the model effectively identifies most of the malicious applications, minimizing the risk of false negatives.

Precision: This metric emphasizes the model's ability to correctly classify predicted positive cases (malicious applications). It is expressed as:

$$\Pr e\,cision = TP/(TP + FP)$$

A high precision value indicates that the model minimizes false positives, preventing unnecessary alarms and system resources being wasted on non-malicious applications.

F1 score: The harmonic mean of recall and precision.

$$F1\ Score = 2 * \frac{Precision*Recall}{Precision+Recall} \qquad \text{Eq (4)}$$

The F1 Score equation is shown in Eq. (4). When precision and recall are both 1, the F1 Score is 1. Only when precision and recall are both strong can the F1 score rise. A more useful metric than accuracy is the F1 score, which is the harmonic mean of recall and precision

Model	Train Accuracy	Test Accuracy	Precision	Recall	F1 Score
Naive Bayes	0.607135	0.605603	0.352030	0.216426	0.249709
Logistic Regression	0.888769	0.862500	0.548490	0.511266	0.529080
Decision Tree	0.987390	0.897845	0.551976	0.563546	0.557122
Random Forest	0.998491	0.948276	0.615365	0.612085	0.612974
AdaBoost	0.896853	0.885345	0.584514	0.518213	0.548417
Extra Trees	0.998491	0.944828	0.612692	0.614741	0.612831
Gradient Boosting Machine	0.953007	0.930172	0.602875	0.595371	0.598768

Table 2: Table of results from the ML models

For Deep Learning Models:

Model	Accuracy	Precision	Recall	F1-Score
Bi-LSTM	0.89568965	0.87503980	0.87055819	0.87232708
LSTM-CNN	0.89310344	0.87934740	0.87178069	0.87413764
Bi-LSTM CNN	0.87974137	0.85537748	0.84986415	0.85239511

Table 3: Deep Learning models results

Three deep learning algorithms were evaluated for their ability to detect malware in Android applications. LSTM-CNN achieved the highest precision of 0.8793474, followed by Bi-LSTM with 0.8750398 .All models outperformed random guessing, with Bi-LSTM CNN showing the lowest accuracy at 0.8797.LSTM-CNN also achieved the best recall and AUC, suggesting its superior ability to correctly identify both benign and malicious applications. These findings highlight the potential of hybrid CNN models for improved Android malware detection and user protection. Further research could explore the performance on larger datasets and investigate the impact of incorporating additional features.

For LSTM-CNN:

- Average False Positive Rate (FPR): 0.026756631623966992
- ROC AUC Score: 0.9398888395819643

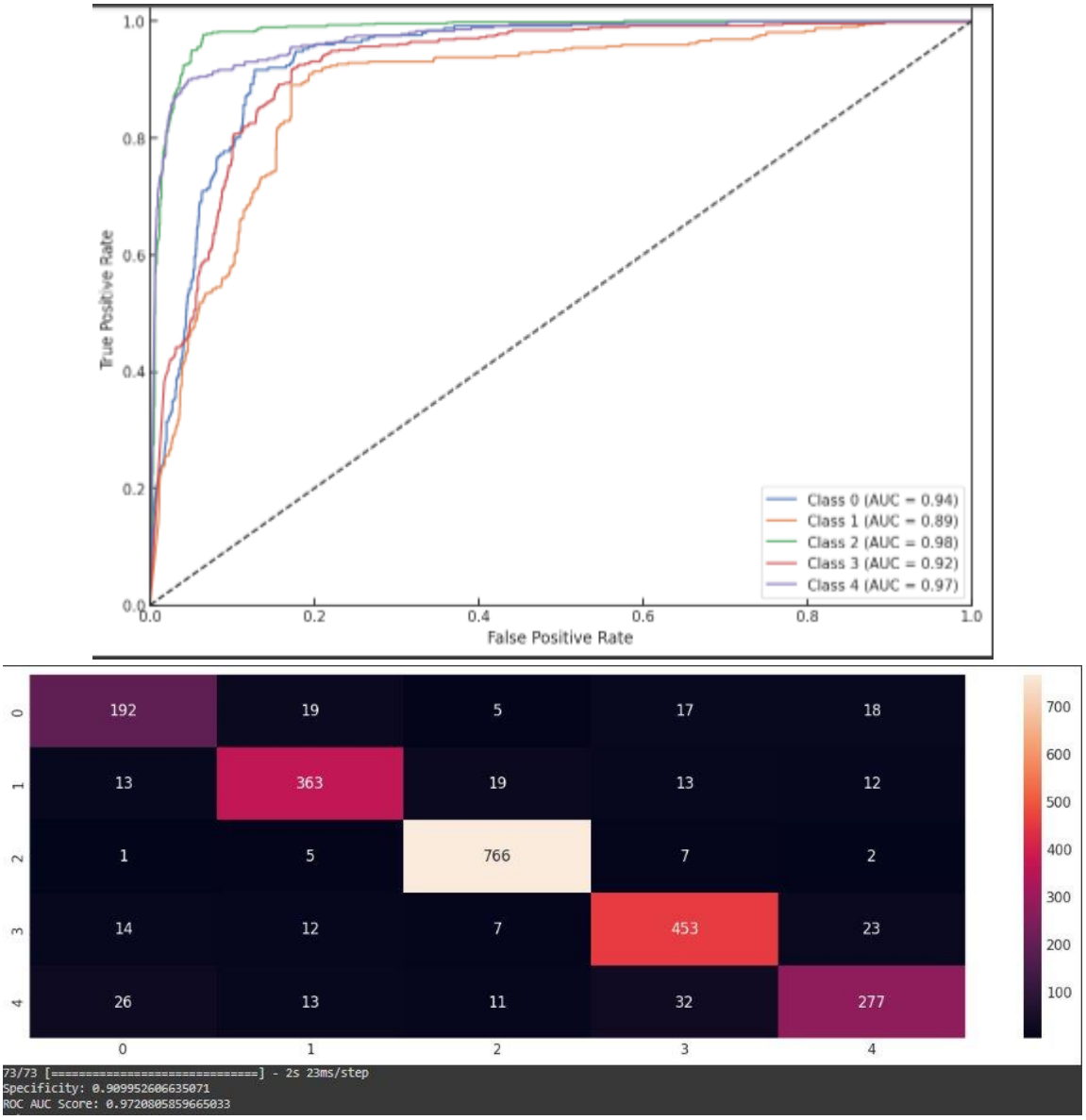

Figure 13: AUC and Confusion matrix for LSTM-CNN model

The Area Under the Curve (AUC) is the measure of the ability of a binary classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the model's performance at distinguishing between the positive and negative classes.

7. Conclusion and Future Enhancements

In this study, we evaluated the efficacy of various machine learning models for early malware detection, with a specific focus on deep learning algorithms. Among these, the LSTM-CNN model emerged as particularly promising, showcasing high accuracy, precision, recall, and F1-score. Its precision of 0.8793474 and recall of 0.87178069 surpassed those of other deep learning models evaluated. Additionally, the LSTM-CNN model achieved the best recall and AUC, indicating its robustness in distinguishing between benign and malicious applications. Given the promising performance of the LSTM-CNN model, we propose employing it as a base model for explainable artificial intelligence (XAI) techniques. By leveraging the strengths of LSTM-CNN and incorporating XAI methodologies, we aim to enhance the interpretability and transparency of our malware detection system. This integration could provide insights into the decision-making process of the model, thereby facilitating better understanding and trust among users and cybersecurity experts. Furthermore, future research endeavors could explore the scalability and generalizability of LSTM-CNN on larger datasets. Additionally, investigating the impact of incorporating additional features, such as behavioral characteristics or network traffic patterns, could further enhance the model's performance in real-world scenarios. Continual learning mechanisms could also be integrated to ensure the model's adaptability to evolving threats, thereby enhancing its long-term effectiveness in mitigating malware risks.

8. Novelty of the solution

The emergence of zero-day malware poses a significant challenge to cybersecurity, as it exploits vulnerabilities before traditional defense mechanisms are established. Unlike known malware, which can be identified using existing mitigation strategies, zero-day malware is characterized by its unprecedented nature and the absence of prior detection methods. Addressing the critical need to identify and mitigate unknown malware threats, our project introduces a novel system that leverages eXplainable Artificial Intelligence (XAI) techniques, specifically SHAP (SHapley Additive exPlanations).

While conventional research has predominantly focused on detecting known malware, our project goes beyond this paradigm to tackle the unique challenges posed by zero-day malware. The proposed system not only detects malicious applications but also strives to provide a deeper understanding of their installation mechanisms. In doing so, it employs advanced XAI techniques, with a specific emphasis on SHAP, to elucidate why certain applications are classified as malware. This approach is groundbreaking in the realm of cybersecurity, where the opaqueness of machine learning models has been a significant concern.

1. Static Analysis with Advanced Shape Modeling (XGBoost/LightGBM):

Novelty: Enhance static analysis through the integration of advanced shape modeling techniques using XGBoost and LightGBM. These gradient boosting models excel at analyzing file structures, permissions,frequency count and intricate code patterns. By leveraging advanced shape modeling, the system gains a nuanced understanding of the structural aspects of malware during static analysis, improving identification accuracy.

2. Dynamic Analysis with Hybrid CNN for Behavioral Modeling:

Novelty: Implement dynamic analysis using a hybrid Convolutional Neural Network (CNN) to model malware behavior. During dynamic analysis, the malware is executed in a controlled environment, and the CNN observes and analyzes its behavior in real-time. The deep learning capabilities of the CNN provide insights

into dynamic features such as system calls, network activities, and runtime behaviors, enhancing the system's adaptability and accuracy.

3. Explainable AI (SHAP) for Interpretability:

Novelty: Integrate SHAP models for explainability. SHAP values provide insights into the contribution of each feature to the model's decision, enhancing transparency in the decision-making process. This ensures that security analysts can understand why a particular decision was made, fostering trust in the model's outcomes and improving interpretability.

Overall Novelty:

- The proposed methodology combines advanced shape modeling with gradient boosting models (XGBoost/LightGBM) for enhanced static analysis.
- Dynamic analysis is enriched by leveraging the behavioral modeling capabilities of a LSTM Convolutional Neural Network.
- The system benefits from explainable AI (SHAP) to enhance transparency and interpretability, ensuring trust in the decision-making process.

Potential Impact:

By harnessing the interpretability power of SHAP, our system offers valuable insights into the decision-making process of the underlying machine learning model. This transparency serves as a cornerstone for building trust and confidence in the accuracy of predictions. Unlike traditional black-box models, our solution ensures that security analysts and end-users can comprehend why a particular application is flagged as malware. This level of transparency is crucial in the battle against zero-day threats, providing actionable intelligence and facilitating prompt response.

Furthermore, considering the dominance of deep learning in various research fields, including malware detection, our project undertakes a comprehensive review specifically focused on the efficacy of deep learning techniques in detecting and classifying zero-day malware. This aspect addresses a significant gap in existing

research, emphasizing the need for tailored approaches to combat the evolving landscape of cyber threats. In summary, our solution not only addresses the urgent challenge of zero-day malware but also introduces a pioneering system that enhances transparency and interpretability through the integration of state-of-the-art XAI techniques.

9. Abstract of Publication

Abstract— In response to the escalating threat of Android malware, this research proposes a hybrid model for early detection and classification using a combination of machine learning (ML) and deep learning techniques. Leveraging Random Forest for feature extraction, the model efficiently processes the complex feature spaces present in the CICMaldroid dataset. Simultaneously, the Long Short-Term Memory (LSTM) CNN, Bidirectional LSTM (BILSTM), Bidirectional LSTM (BILSTM) CNN are employed for further analysis. The proposed approach integrates predictions from both the machine learning and deep learning models to achieve comprehensive results. Data preparation involves outlier removal and Synthetic Minority Over-sampling Technique (SMOTE) for class imbalance. Random Forest analysis guides feature selection, optimizing the model's efficiency. Cross-validation ensures robustness, and Explainable AI techniques, specifically SHAP, enhance interpretability. Experimental results demonstrate the model's effectiveness, achieving high overall accuracy and specific identification of Android malware types. The proposed methodology provides a streamlined and transparent approach to early malware detection, contributing to a deeper understanding of the decision-making process behind each prediction.

11. References/Bibliography

[1] A. R. Nasser, A. M. Hasan, and A. J. Humaidi, "DL-AMDet: Deep learning-based malware detector for android," Intelligent Systems with Applications, vol. 21, p. 200318, 2024.

[2] Munyeong Kang, Jihyeo Park, Seonghyun Park, Seong-je Cho, and Minkyu Park. 2021. Android Malware Family Classification using Images from Dex Files. In The 9th International Conference on Smart Media and Applications (SMA 2020). Association for Computing Machinery, New York, NY, USA, 181–186.

[3] M. Chen, Q. Zhou, K. Wang, and Z. Zeng, "An Android Malware Detection Method Using Deep Learning based on Multi-features," in 2022 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), pp. 187-190, 2022.

[4] S. Amenova, C. Turan, and D. Zharkynbek, "Android Malware Classification by CNN-LSTM," in 2022 International Conference on Smart Information Systems and Technologies (SIST), pp. 1-4, 2022.

[5] Y. Fang, Y. Gao, F. Jing and L. Zhang, "Android Malware Familial Classification Based on DEX File Section Features," in IEEE Access, vol. 8, pp. 10614-10627, 2020, doi: 10.1109/ACCESS.2020.2965646.

[6] Hani AlOmari, Qussai M. Yaseen, Mohammed Azmi Al-Betar, A Comparative Analysis of Machine Learning Algorithms for Android Malware Detection, Procedia Computer Science, Volume 220, 2023, Pages 763-768, ISSN 1877-0509.

[7] M. Al-Fawa'reh, A. Saif, M. T. Jafar and A. Elhassan, "Malware Detection by Eating a Whole APK," 2020 15th International Conference for Internet Technology and Secured Transactions (ICITST), London, United Kingdom, 2020, pp. 1-7.

[6] H. Bai, G. Liu, W. Liu, Y. Quan, and S. Huang, “N-gram, semantic-based neural network for mobile malware network traffic detection,”

Security and Communication Networks, vol. 2021, pp. 1–17, 2021. doi: 10.1155/2021/5599556.

[7] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, “Toward developing a systematic approach to generate benchmark android

malware datasets and classification,” in 2018 International Carnahan Conference on Security Technology (ICCST), 2018, pp. 1–7. doi:

[8] R. S. Arslan, “Androanalyzer: Android malicious software detection based on deep learning,” PeerJ Computer Science, 2021.

[9] Z. Ma, H. Ge, Z. Wang, Y. Liu, and X. Liu, Droidetec: Android malware detection and malicious code localization through deep learning,

2020. doi: 10.48550/ARXIV.2002.03594. [Online]. Available: https://arxiv.org/abs/2002.03594.

[10] Y. Fang, Y. Gao, F. Jing, and L. Zhang, “Android malware familial classification based on dex file section features,” IEEE Access, vol. 8,

Reference Links:

https://doi.org/10.1145/3426020.3426069

https://ieeexplore.ieee.org/document/8955840

https://doi.org/10.1016/j.procs.2023.03.101

https://doi.org/10.23919/ICITST51030.2020.9351333

https://doi.org/10.1109/ITOEC.2018.8740537

https://www.sciencedirect.com/science/article/pii/S2667305323001436

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9844642

https://ieeexplore.ieee.org/document/9945816