RESEARCH ARTICLE

ViTDroid and Hybrid Models for Effective Android and IoT Malware Detection

Inam Ullah Khan[1], **Asim Zeb**[2,*], **Taj Rahman**[1], **Fida Muhammad Khan**[1], **Zeeshan Ali Haider**[1] **and Hazrat Bilal**[3,4]

[1] Department of Computer Science, Qurtuba University of Science & Information Technology, 25000 Peshawar, Pakistan
[2] Department of Computer Science, Abbottabad University of Science and Technology, Abbottabad 22010, Pakistan
[3] College of Mechatronics and Control Engineering, Shenzhen University, Shenzhen 518060, China
[4] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China

Abstract

This paper introduces ViTDroid, a novel hybrid model that combines Vision Transformers (ViTs) and recurrent neural networks (RNNs) to enhance Android and IoT malware detection. ViTDroid addresses critical challenges by leveraging ViTs to capture global spatial dependencies and RNNs (LSTM and GRU) to model temporal patterns, enabling comprehensive analysis of complex malware behaviors. Additionally, the model integrates explainability tools, such as LIME and SHAP, to enhance transparency and trustworthiness, essential for real-world cybersecurity applications. The study evaluates ViTDroid's performance against conventional models, including RNN, LSTM, and GRU, using accuracy, precision, recall, and F1 score as evaluation metrics. Results demonstrate that ViTDroid achieves superior performance with an accuracy of 99.1% for Android malware and 98% for IoT malware. Precision and recall values reach 0.99 and 0.98, respectively, for Android, and 0.97 and 0.98 for IoT, with F1 scores of 0.99 for Android and 0.97 for IoT. These findings underscore ViTDroid's potential as a robust, efficient, and explainable solution to combat evolving threats in mobile and IoT ecosystems, paving the way for future advancements in malware detection systems.

Keywords: Android malware, IoT malware, RNN, LSTM, GRU, ViTDroid, hybrid models, malware detection, deep learning.

1 Introduction

The rise of Android and IoT devices has brought transformative technological benefits, but also heightened cybersecurity risks, with sophisticated threats like polymorphic and metamorphic malware bypassing traditional defenses. Vision Transformers (ViTs), initially developed for computer vision, have recently shown promise in malware detection by capturing global dependencies in opcode sequences through self-attention mechanisms.

Unlike CNNs, which focus on local features, ViTs excel

Submitted: 06 January 2024
Accepted: 10 March 2024
Published: 29 March 2024

Vol. 1, **No.** 1, 2024.
10.62762/TACS.2024.521915

***Corresponding author:**
✉ Asim Zeb
asimzeb1@gmail.com

Citation
Khan, U. I., Zeb, A., Rahman, T., Khan, F. M., Haider, Z. A., & Bilal, H. (2024). ViTDroid and Hybrid Models for Effective Android and IoT Malware Detection. *IECE Transactions on Advanced Computing and Systems*, 1(1), 32–47.

in identifying long-range relationships within data, making them particularly effective for detecting subtle, malicious patterns. Recent advancements integrate ViTs with recurrent models like LSTM and GRU to enhance temporal dependency modeling, offering a robust solution for evolving malware threats.

This study introduces ViTDroid, a hybrid model combining ViTs and RNNs, achieving high accuracy for Android and IoT malware detection. Additionally, explainability tools like LIME and SHAP ensure transparency, making ViTDroid a trustworthy and advanced solution for modern cybersecurity challenges.

The rapid advancement of mobile and IoT devices has profoundly impacted both personal lives and business operations, revolutionizing human interaction with technology. However, this progress has also introduced significant and often uncontained risks to cybersecurity. In particular, smartphones, especially portable handheld communication devices running operating systems like Android, have become ubiquitous, making the need for robust security measures more critical than ever [1, 2]. Although present smartphone users are more than 6.5 billion and projecting that IoT devices will be more than 75 billion by 2025, the severity and vulnerability of cybersecurity threats linked to these technologies is growing rapidly. Many of these devices are targeted by well-crafted malware to take advantage of the weaknesses of both the software and the physical hardware of the devices. As the most popular OS on which many devices are produced, Android-based devices are invaders exposed to every type of malware, from Trojans to ransomware [3]. Consequently, the Internet of Things – connected devices ranging from household to industrial equipment- lack sufficient safeguards; these devices can be hacked to threaten privacy and disrupt networks [4].

Vision Transformers (ViTs) are advanced deep learning models designed to capture global dependencies in data using self-attention mechanisms. Unlike traditional convolutional neural networks (CNNs), which focus on local features, ViTs analyze malware opcode sequences as image-like patches, enabling effective detection of subtle and distributed malicious behaviors. Explainable AI (XAI), on the other hand, focuses on making machine learning models interpretable and trustworthy. Tools such as LIME (Local Interpretable Model-Agnostic Explanations) and SHAP (SHapley Additive exPlanations) identify specific features or patterns influencing the model's predictions. In the context of cybersecurity, XAI enhances the usability of models by enabling experts to understand and validate outputs, ensuring reliable and transparent decision-making.

Most conventional tools for identifying malware include signature detection, when the system searches for already known malware patterns. These approaches, though viable for identifying recognized threats, are not helpful when it comes to emerging threats. Polymorphic malware, the kind of threat that changes its appearance to avoid identification, and metamorphic malware, which can rewrite its code entirely, are threats to which signature-based systems cannot respond [5]. Heuristic-based methods, which work by analyzing the behavior of applications or files in an attempt to determine whether it comprises threats, provide some enhancement but still suffer from several difficulties in sensing the more persistent new generation of malware that is designed to look like normal user activities or that employ techniques that would try to dodge detection [6]. Therefore, an increasing interest has appeared in using ML and DL approaches for malware detection. All of these methods are well suited to the modern threat landscape because they can learn from large data sets, incorporate newly discovered threats, and recognize subtle patterns that may be missed by traditional methods [7].

Numerous machine learning approaches, deep learning models– including RNNs, LSTM, and GRU, have been used often to identify malware, especially within sequential data sources like system calls, APIs, and network traffic logs. RNNs, which are the basic structures of deep learning for sequence learning based on time series analysis, should directly apply to tasks such as malware behavior detection because the order and time interval of actions are important for the task [8, 11]. However, the problem with such RNNs is that they are prone to vanishing gradients, making it hard for them to learn from very long sequences, particularly for complicated malware[9]. This is a major drawback when using RNNs to perform inference as an input item can only influence distant targets, not the entire sequence. This issue is overcome in LSTM networks as they have memory cells in which they can store information into longer times, making them appropriate for long-range dependencies in a sequence[10, 12]. GRUs, another type of RNN, present fewer parameters than LSTMs but are known to provide nearly the same performance in capturing temporal relations. These sequence-based models have been demonstrated to be useful in detecting malware by patterns of behavior that stretch over a long time range, for example, intrusion of APIs or network traffic that is anomalous over time. RNNs, GRUs, and LSTMs work well with sequential data, whereas more complex, multiple-dimensional data like images, graphs, or well-structured sensor data, which is more common in today's mobile and IoT devices, are better handled in this architecture. This is where Vision Transformers (ViTs) come into play La-Vie. Originally designed for computer vision problems, ViTs defined high performance in terms of transforming the data through spatial and contextual relationships through the attention mechanism [13, 14]. Such versatility of the self-attention mechanism implemented by ViTs enables them to ignore the spatial relationships of features and focus on the critical features present within input data this results in the model's capability to identify various patterns and outliers within data. Recent work has also extended the scope of ViTs beyond conventional image recognition-related tasks and provided evidence that it can be used for time-series analysis and potentially designing anomaly detectors. In this work, we investigate the performance of not only deep machine learning models but also ViTDroid, a Vision Transformer model for malware detection, in parallel to conventional models such as RNN, GRU, and LSTM. As a result of incorporating Vision Transformers in both time and space detection, ViTDroid provides a comprehensive way of precisely detecting a range of malicious software for Android and IoT devices. First, as mentioned above, while there are several traditional sequence-based models for learning, such as LSTMs, for detecting malware with sequential behavior patterns, the proposed ViTDroid, on top of maintaining these features further comes equipped with attention capabilities, which can explore contextual relationships in the data, which is highly important for detecting seemingly less differentiated yet actually extremely significant patterns of behavior in malware.

Besides the detection capabilities of these models, this research proposed the approach of the explainability of machine learning in cybersecurity applications [15]. A sophisticated model like ViTDroid, built using deep learning and attention mechanisms, is often called a 'black box,' implying that it is difficult to explain the inner computations by which it made a particular decision. In cybersecurity, where decisions have to be trustworthy and timely, opacity about how such a model came to its prediction can be an inhibitive factor. This is especially so given the serious consequences that may come with either incorrect identification of malware or failure to identify the malware. This is why techniques of explaining how and why particular predictions were made, known as Explainable AI (XAI), are crucial. Here, in this study, we want to test the efficiency and applicability of ViTDroid and other models that may be developed or proposed andconsider the possibility of explaining such models and making model interpretability more comprehensible to cybersecurity experts.

Since the threats are constantly changing in nature the detection systems should also be capable of detecting both known and unknown threats. Lastly, with the proposed Vision Transformers in ViTDroid, the merger of Vision Transformers and sequence-based models is a good direction for future Malware detection systems. This research offers inspirational results of the extensive analysis of the performance of the range of deep learning models for Android and IoT malware classification. It opens the perspective to develop new advanced hybrids of the deep learning models to enhance the detection rate and counteract the increasing threats of current cyber attacks [16][17]. Lastly, incorporating explainable AI techniques adds more onus to making these models useful in real-life cybersecurity settings, making it possible to get better and more transparent malware detection models.

The overall findings of this research demonstrate that the availability of high-level data machine learning and deep learning models, including ViTDroid, can improve the sphere of malware detection for both mobile and IoT platforms. This is because ViTDroid harnesses the power of Vision Transformers and at the same time benefits from conventional models like LSTM in detecting patterns in data that are used in presenting advanced malware threats. The work also focuses on the interpretability of the model to guarantee its efficacy in practical cybersecurity use when deployed by such systems. As the threat landscape advances, resources such as ViTDroid may serve as the main vehicle for defining the future of detection technologies and malware prevention architectures.

- This work proposes a superposition of Vision Transformers (ViTs) input/recurrent Recurrent Neural Networks (RNNs- LSTM/GRU) to take both spatial and temporal relations in opcode sequences that will improve the accuracy of the Android and IoT malware discovery.

- Ensemble learning techniques (boosting and bagging) are integrated into the ViTDroid model to improve its robustness and accuracy, addressing issues of data imbalance and detecting previously unseen malware attacks.
- Advanced attention mechanisms and model-agnostic interpretability tools (LIME and SHAP) are incorporated to make the malware detection model more transparent, allowing experts to understand and trust the model's predictions.

2 Background

The morphological changes observed in mobile devices and IoT system malware further complicate the detection and protection of these malware, especially in the context of smart cities. The evolution of inadvertent complex malware variants undermines conventional cybersecurity strategies, therefore proposing incorporating deep learning technologies. This chapter provides an overview of the background of some of the important areas pertinent to the study: the malware detection methods, the use of deep learning models such as Vision Transformers (ViT), and a hybrid model implementation to boost the detection of malware, as well as the approaches of making untangled and more robust models in the framework of Android and IoT malware.

2.1 In Mobile and IoT Devices Malware Detection

Malware in mobile and IoT has emerged as a new threat to cybersecurity mainly because these devices are increasingly used in sensitive areas [18–20, 31]. Subsequently, smartphones, especially those with the Android platform, have become the largest opportunities for attackers, given their ubiquity, availability, and versatility. Traditional malware detection methods, such as signature-based and heuristic approaches, struggle with modern threats like polymorphic and metamorphic malware, which evade detection through obfuscation and dynamic changes. While machine learning models like RNNs, LSTMs, and GRUs improve sequential data analysis [21], they often fail to capture global dependencies and lack explainability, limiting their effectiveness. The number of apps and user actions is continually growing, which, along with other factors, forms an unbounded space for activity, against which it is challenging to defend using traditional approaches based on signatures [22].

Like PCs, IoT devices used today in smart cities for infrastructure monitoring, utility management, and other improvements in people's daily lives are also easily infected by malware. These devices are resource-constrained, often have weak security controls in place, and are members of large complex networks that are challenging to monitor [23]. These devices' flaws can be used to launch Distributed Denial of Service (DDoS) attacks, data theft, or incapacitating essential services such as healthcare and traffic systems [24]. Traditional malware detection techniques, such as signature-based detection, work by recognizing known signatures of malware. Although helpful for inherent attacks, these techniques fail to identify zero-day attacks, polymorphic malware, or a new breed of viruses [25]. For this reason, new techniques to detect these malware classifications that can be applied effectively to future threats are also required. This is true for most ML and DL models that can easily capture data patterns related to malicious activity even if the threats are unknown or in a new variant [26].

2.2 Vision Transformers (ViTs) and Their Application to Malware Detection

Designed to focus on vision, Vision Transformers (or ViTs) are a relatively new addition to deep learners and have demonstrated impressive performance in image classification. The idea that forms the basis for ViTs is to partition an image into smaller segments and process each segment or patch as an individual token. Making tokens of ViTs attend to each other across space also helps ViTs model long-range dependencies in data; it suggests that ViTs are particularly suitable for tasks where the global context is crucial, such as object recognition in images. ViTs have recently been applied to the field of malware detection since they provide the opportunity to capture global dependencies in sequential data, such as opcode sequences originating from Android and IoT malware. Unlike the original Convolutional Neural Networks (CNNs), which extract local spatial features by convolutions, ViTs regard the whole input data as a sequence and the interaction between all patches (or tokens) simultaneously. This allows ViTs to analyze local dependencies in the data and index important patterns of malicious behavior that might go unnoticed at the granular level [27, 28]. This article shows that ViTs have one crucial benefit over other architectures—their ability to capture long-range dependencies. As seen regarding malware analysis, malicious activities usually extend through several instructions, and identifying such activities means recognizing the interactions between far opcodes. Unlike most embeddings that operate on a single opcode, ViTs work on a sequence of all or at least a subset of them, allowing ViTs to detect trop global that characterizes malware behavior [29]. Some recent works have shown that ViTs are more effective than the traditional CNN-based counterparts in certain malware detection [35] tasks because of their better ability to capture these relations [30]. Nevertheless, ViTs have come across some issues when working with sequential data, and more specifically, with opcode sequences. ViTs can capture global dependencies but do not capture temporal dependencies between the opcodes, which this problem heavily relies on due to sequential processing performed by malware. This has been done due to the limitations of the ViTs and the subsequent investigation of more powerful models, such as Recurrent Neural Networks (RNNs), that are effective for sequential data.

2.3 Hybrid Models

Further, the RNN utilizes the ViTs to identify the presence of malware or not. The integration of Vision Transformers (ViTs) and Recurrent Neural Networks (RNNs), namely Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), becomes a natural solution to avoid the problems that occur with each model. Although ViTs consider global patterns of the data, RNN memorizes previous inputs' information, which makes them suitable for understanding the sequence of indices in the malware operation.

LSTMs are a subtype of RNN machines that perform exceptionally well with long-term sequence dependencies. Gating mechanisms help to protect from the vanishing gradient problem, which tends to happen if traditional RNNs are used with long sequences by allowing the model to decide which information to remember and which to forget at each step. This makes LSTMs suitable for analyzing dependencies on the timelines of actions in malware samples, given that the temporal order of operations defines malicious behavior. LSTMs are similar to GRUs but allow the data to be updated at each time step, which makes them more computationally extensive but, at the same time, they can get at least as many parameters capturing long-range dependencies in sequential data. The further hybrid models can describe the malware's behavior better, Created with ViTs that deal with global dependencies and LSTMs or GRUs that address sequential ones. It combines this to enhance the effectiveness of the tuple detection model as the model needs to consider the order in which these operations are called when determining the general structure of the malware.

Recent works show that the integration of ViT and RNN has been pioneered in many fields and domains, ranging from time series to natural language processing domains where both spatial and temporal information are essential. In malware detection this approach has proved effective, enabling models to be developed that capture the richness and the multifaceted nature of malware.

2.4 Ensemble Learning for Malware Detection

Ensemble learning is one of the strong concepts in machine learning in which different models are used jointly to enhance the characteristics of the decision. Finally in ensemble methods such as bagging and boosting different individual models are built using different sections of the data or else the same section of data is used by building them using different algorithms and the final decision is constructed from the individual model's decisions. The strength of ensemble methods is that they lessen the problem of overfitting and increase the model generalization since a number of models are used. Smoothing methods like Ada Boosting and Gradient boosting are used where the learning algorithm builds a sequence of models each demonstrating different errors. This they achieve at the expense of a slight slowdown in computational speed and the emergence of higher order polynomials, and a strong learner that can perform well even on complex and noisy data. The rgenetic algorithm [36, 37], involve the construction of a set of models trained on different subsets of the data, and making a decision based on the response of all of the models. Both techniques are more suitable for dealing with imbalanced datasets frequently used in malicious code detection when the number of normal samples is significantly greater than that of malicious samples.

Specifically to the detection of malware, the ensemble learning method proposed in this paper can enhance the performance of each component model which is easily affected by variance and bias when they face complex and unbalanced datasets. As the name suggests ensemble methods can also improve the models' resistance to adversarial attacks or previously unknown strains of malware. In the proposed

ViTDroid model, the idea of ensemble learning can be implemented on top of the hybrid ViT-RNN model to enhance the malware detection result's reliability and accuracy.

2.5 Explainability in Malware Detection

Interpretability in the context of a Machine learning model means how easily the decisions made by a model can be explained. In the case of cybersecurity, the crucial steps of identifying that certain software is either malicious or benign are taken by automated systems, thus the respective decisions need to be explainable. This is especially true when deep learning models ViTs and hybrid models are used as most of them are considered "Black Box" models it is difficult to understand their internal mapping mechanisms.

As for this issue, several approaches have been made to enhance the interpretability of deep learning models. Some of them include attention mechanisms, for instance, enabling the models to give priority to portions of the input data, which are most important in making the forecast. In malware detection, attention mechanisms can identify which areas of the opcode sequence are more indicative of malicious activity, thus making the model's decision-making more interpretable.

Furthermore, non-specific interpretation techniques, such as LIME (Local Interpretable Model-Agnostic Explanations) or SHAP (Shapley Additive exPlanations) can generate explanations for the predictions made by a diverse model. LIME relies on the concept of substituting a complex model with a simpler model, to explain the predictions made for an individual instance while SHAP uses concepts of cooperative game theory to explain the level of contribution provided by each feature to the final prediction. These tools enable a user to know why a particular sample has been categorized as either malicious or benign, which is helpful in trust and validation in cyber-security applications [38, 39].

Complementing the ViTDroid model with attention mechanisms and interpretability tools, this research improves the model's trustworthiness and supports the safe utilization of the model's predictions by cybersecurity professionals.

2.6 Data Augmentation and Transfer Learning for Coping with Data Poverty

A major limitation typical of developing machine learning models for malware detection [32–34] is the availability of labeled data, particularly for new or relatively rare types of malware. This problem can be solved to a certain extent using a data augmentation technique which makes the amount and the variety of data in the training set larger artificially. As mentioned before, common augmentation in image classification includes rotation, scaling, and flipping among them. When dealing with opcode sequences data augmentation can be performed by changing the structure as well as the sequence of opcodes but still pointing toward an evil purpose. This helps the model to generalize more and hence be made more immune to other unseen samples.

Another powerful technique that also corrects the existence of scarce labeled data is known as transfer learning. When deep learning models are trained from scratch, they require more datasets and a longer time to obtain the result; conversely, pre-trained deep learning models also make use of learned features from related tasks or domains. In malware detection, transfer learning [40–42] has been found effective when the original models from related large datasets from other environments or any general image classification datasets have been used. This enables the model to harness the knowledge that is embedded in the pre-trained model even under conditions where the only available data is labeled.

3 Proposed Architecture

In this section, we present the architecture of ViTDroid, an advanced deep-learning model designed for Android and IoT malware detection, as shown in Figure 1. The proposed model integrates Vision Transformers (ViTs) with hybrid architectures to enhance its ability to detect and classify malicious behavior in opcode sequences and system calls. This chapter details the key components of ViTDroid, including the input representation, model architecture, hybrid components, training process, and the techniques used to enhance detection accuracy, explainability, and robustness.

3.1 Overview of ViTDroid Architecture

ViTDroid is designed to address the limitations of traditional machine learning models in malware detection by leveraging the power of Vision Transformers for capturing global dependencies within opcode sequences. The architecture of ViTDroid combines the strengths of ViTs with Recurrent Neural Networks (RNNs), such as Long Short-Term Memory (LSTM) networks or Gated

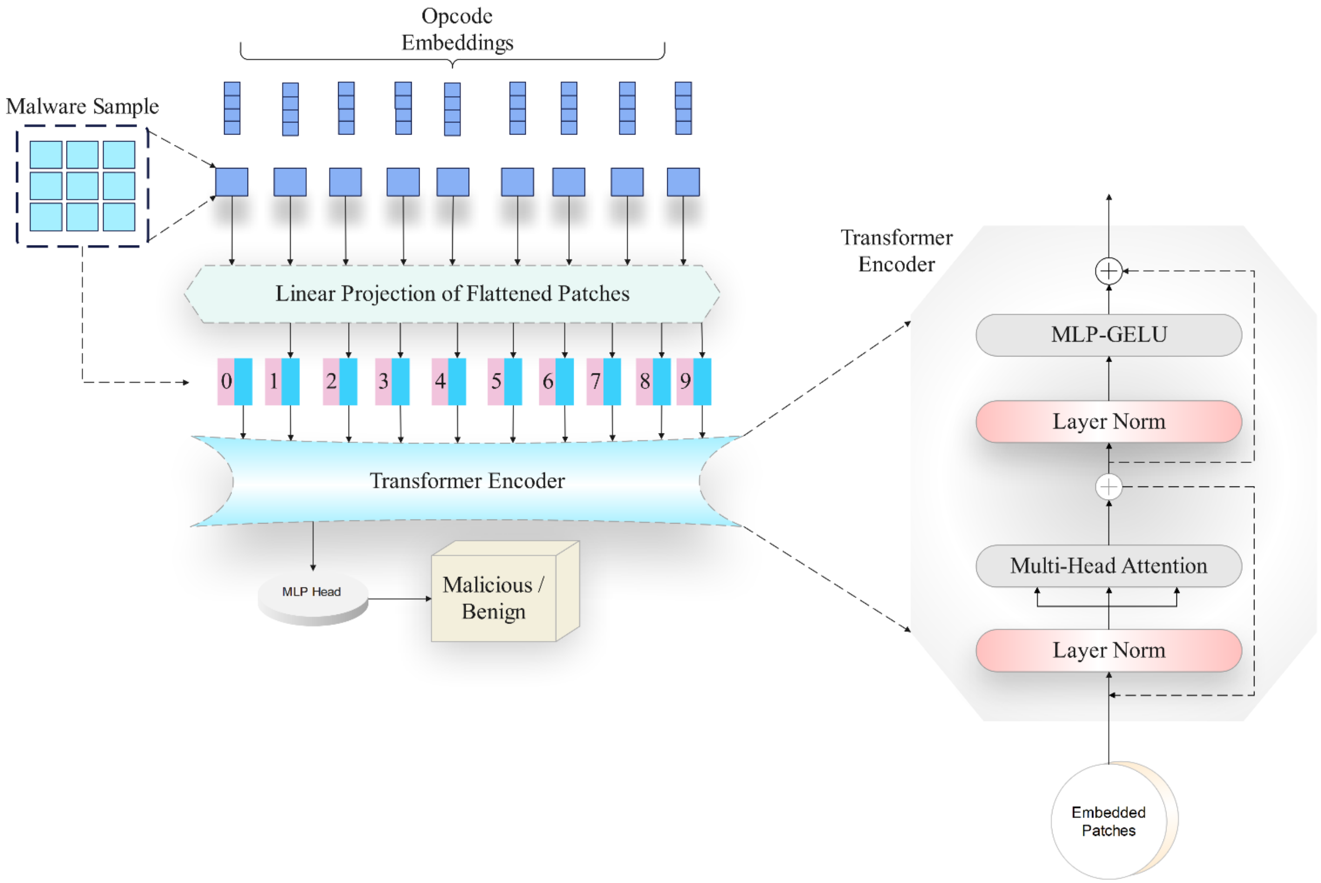

Figure 1. Proposed architecture.

Recurrent Units (GRUs), to model both the spatial and temporal dynamics of malware execution. ViTDroid combines Vision Transformers (ViTs) and recurrent neural networks (RNNs) to enhance malware detection by leveraging both spatial and temporal dependencies. Malware opcode sequences are tokenized into patches and processed by the Vision Transformer module to capture global dependencies through self-attention mechanisms. The RNN module, using LSTM or GRU, analyzes sequential relationships to identify temporal patterns of malicious activity. Outputs from these modules are integrated using ensemble techniques like boosting and bagging to improve robustness and accuracy. Additionally, explainability tools such as LIME and SHAP enhance transparency by highlighting features that influence predictions, ensuring trust and usability in cybersecurity applications.

At a high level, the ViTDroid architecture consists of several components:

- **Input Representation:** Before they are fed into the deep learning model the raw opcode sequences extracted from a set of Android and IoT malware samples are preprocessed and transformed. The second step is the opcode sequence to be tokenized, followed by the division of opcode sequences into segments or patches in the form of input to the Vision Transformer module.

- **Vision Transformer Module:** The basis for ViTDroid is the Vision Transformer that treats the input opcode sequences as image-like patches as well as maintaining global dependencies across the entire sequence.

- **Hybrid RNN Module:** To capture temporal dependencies within the opcode sequences, ViTDroid employs a recurrent module (LSTM/GRU) that captures the sequential interdependencies between opcodes.

- **Ensemble Learning:** To enhance the accuracy and the optimum robustness, ensemble learning is used on the output of the Vision Transformer and the RNN module.

- **Explainability Layer:** ViTDroid also includes the understanding of attention and the use of LIME and SHAP for making the predictions transparent.

3.2 Input Representation: Opcode Sequences as Patches

To fit the input data into the Vision Transformer, the raw opcode sequences are transformed into patches similar to the image ones. Each opcode is taken as a feature and a window of a fixed size is applied by sliding it over the sequence to produce patches. These patches are then inserted into the fixed length vectors using the embedding layer showing the model what to learn in terms of opcode sequences features.

This conversion allows the Vision Transformer to learn over opcode sequences similar to how it learns over an image where patches are local fragments of the sequence and learn dependencies over the entire sequence by attending to distinct patches simultaneously. It is specifically beneficial for malware detection since the model can gain both the detailed local structures (local opcodes) and the global context (coarse patterns across opcodes) in the malware execution sequence.

3.3 Vision Transformer Module: Global Dependency Capture

The Vision Transformer (ViT) module is the central module of the ViTDroid. In ViTs, the patches focused on capturing global dependencies in the input data which is important in observing the behavior of malware across different instructions or opcodes. It is well known that convolutions are used to learn local features in a standard convolutional model, but this hinders the performance of learning long-range dependencies. To this, end, ViTs do not process the sequence one element at a time; rather, the whole sequence's interaction is captured.

When it comes to ViTDroid, the ViT module learns dependencies across the entire sequence acting on the embedded opcode patches in parallel. This makes it possible for the model to identify other obscure characteristics that reveal the violation of malicious behavior not defined by the local level Opcode. For the same reason, the self-attention mechanism used in ViT models enables an adjustment of weights by which the data will be weighted in a way that is beneficial to focus on important patches while disregarding less relevant ones.

3.4 Hybrid RNN Module: Temporal Dependency Modeling

Although ViTs are excellent in modeling dependencies across the entire globe, malware can have intricate temporal patterns, and any reordering of opcodes that a malware might induce is crucial to identifying its misbehavior. To overcome this, in ViTDroid, there is a combined module, where there is Recurrent Neural Network (RNN) – Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU). These above-described RNN variants are built to handle sequential data and learn temporal patterns by having a memory of earlier inputs. Malware detection requires both spatial analysis of opcode relationships and temporal analysis of sequential behaviors. Vision Transformers (ViTs) capture global spatial dependencies, identifying subtle malicious patterns, while RNNs like LSTM and GRU excel at modeling temporal relationships critical for detecting chronological malware behaviors. ViTDroid combines these strengths, enabling comprehensive detection of complex malware by addressing both spatial and temporal challenges effectively.

3.5 Ensemble Learning: Improving Accuracy and Stability

In a bid to enhance the distinct and consistent outcomes of ViTDroid, boosting and bagging are combined learning methods that are integrated alongside the tool. Voting operates by taking several models and using them in an ensemble, as the inconsistency and prejudice problems hurt performance. In ViTDroid, such techniques as Gradient Boosting or AdaBoost are used, where in the process of learning several models are trained, and each subsequent one is aimed at improving results on the problems that failed the previous model. This aids in modifying the model to extremely function better in response to sophistical and noisy data inputs.

Further, some modifications like bagging techniques like Random Forests may be applied for parallel use with Vision Transformer and RNN sub-modules. The results are combined by training several models on various portions of the data and then making the consolidated prediction. This ensemble approach decreases the probability of over-fitting and increases the likelihood for ViTDroid to perform well on completely unseen cases, thus making the method more resistant to adversarial attacks and other modifications of the used samples of malware.

3.6 Explainability Layer: Enhancing Model Transparency

Cybersecurity is a domain where it is very important to know how a given model decides that a sample is malicious or benign, especially in the case of inspecting the samples that are classified as malware. To promote transparency of the results, ViTDroid has components of explainability that make the public trust in the output. These mechanisms are used to focus on the specific sections of the opcode sequence which might have a strong bearing on the model's decision. This made it more convenient to shift the focus of security analysts on the special opcodes or patterns that led to the detection of malware reducing the interpretative part of the model.

To improve the general interpretability ViTDroid uses post hoc interpretability tools such as LIME and SHAP. These tools help to determine how important each given feature contributes to the model in question (in this case, each opcode or patch). Thus, the applications of these tools, ViTDroid can help the security professionals with understandable standpoints on why such behaviors are malicious, so that those security professionals can make the right decision on the suspicion.

3.7 Training Process: Data Augmentation and Transfer Learning

To demonstrate that ViTDroid achieves good performance on practical datasets the training is done using modern approaches such as data augmentation and transfer learning. Data augmentation is used because, for malware detection, there are few samples available that are labeled for training. Thus, by creating the new training samples based on the transformations (for example the permutation of the opcodes order) the model can generalize upon the new, unseen types of malware and does not get overtrained on certain variants.

Transfer learning is also used to use models ready on related tasks, for example, image classification or malware detection in other contexts. This approach helps ViTDroid leverage the knowledge embedded in pre-trained models, and perform feature extraction to the particular task of Android and IoT malware classification. The process of transfer learning is most useful when training with a relatively small number of labeled inputs because the model itself can learn more quickly from small sets.

4 Experiments and Results

In this section we outline the experiments conducted, the evaluation metrics used, and the datasets employed to test the ViTDroid model for detecting malware on Android and IoT devices. For this purpose, all our experiments are designed to analyze the number of detections of the ViTDroid architecture regarding the high accuracy, robustness, and explainability of the results. We evaluate ViTDroid against conventional malware detection models and establish the contribution of several developments including Vision Transformers, hybrid Recurrent Neural Networks, and Ensemble learning.

4.1 Experimental Setup

To assess the performance of ViTDroid we used a set of Android and IoT malware samples and the main focus was on smart city IoT devices. The datasets were then cleaned and tokenized to ensure opcode sequences were converted into image patches for the Vision Transformer module. Furthermore, the same sample of malware was split into training and test data sets to measure its generalization capability.

We trained and tested fusing hardware and software environments into one training and testing environment. The experiments were performed on a machine installed with an NVIDIA GPU, which provided resources for training complicated deep-learning models. The model was reproducibly built and experimented on with the help of established deep learning frameworks such as TensorFlow and PyTorch.

4.2 Datasets

To assess the execution of ViTDroid, two separate sets of malware were used, the first set was from Android malware, and the second set was from IoT malware. We chose these datasets to prove that the model works universally focusing on the Android domain and IoT devices in smart cities at that.

4.2.1 Android Malware Dataset

The dataset of Android malware used in this work is a collection of the APK samples of the malevolent applications, taken from authoritative stores or received from well-known malware archives. Android malware in the dataset comes from different families depending on the type of behavior and these include; trojans, ransomware, spyware, adware, and other malicious apps.

The key components of the Android malware dataset include:

- **Opcode Sequences:** These sequences are obtained from the bytecode of APK files which gives the sequence of instructions carried out by the Android app. It is important to note that each opcode is associated with a single operation in the functioning of the Android application.

- **System Call Traces:** It must be noted that there are also records of system calls invoked by the malware at runtime within the dataset. It is also able to record such interaction that includes file reading, network calls, and system resource use by the app.

The opcode sequences extracted from the programs were first tokenized and transformed into patches of fixed size and fed into the Vision Transformer model. The samples within the current dataset were categorized into two classes: malicious and benign thus offering the ground truth for training and evaluation.

4.2.2 *IoT Malware Dataset*

The IoT malware dataset comprises samples gathered from simulated smart city IoT devices in the real world. These devices include environmental sensors, smart meters, surveillance cameras, and industrial control systems. Since IoT devices are connected and hence are very vulnerable to attackers & have poor security the dataset included all the centrality of IoT malware.

The key components of the IoT malware dataset include:

- **Opcode Sequences:** Like the Android dataset, the IoT malware dataset holds the execution traces of malware samples with vulnerability exploitation in IoT devices. These traces reflect the minor tasks performed by the nasty payloads as soon as they penetrate a device.

- **System Call Traces:** Besides opcode sequences, system call traces were also collected to analyze how malware triggers the underlying operating system of IoT devices. These traces offer an understanding of malware conduct including data leaking, elevation of access privileges, or unauthorized management of IoT.

- **Dataset Limitations:** The datasets used, comprising Android and IoT malware samples, have limitations that may impact generalizability. While diverse malware types are included, they may not fully represent emerging threats. The IoT dataset, partially generated in simulations, may lack real-world complexity. Balanced class distributions simplify training but do not reflect real-world imbalances, and potential geographical or contextual biases may limit broader applicability. Future work should expand dataset diversity, include real-world IoT samples, and evaluate performance on imbalanced and region-specific datasets to improve robustness and adaptability.

The IoT malware dataset was also preprocessed in the same way as the Droid with the use of opcode sequences and system calls to patches which will be input to ViTDroid. To maintain balanced data, equal numbers of benign and malicious samples were taken.

4.2.3 *Data Preprocessing*

The preprocessing steps for both datasets are as follows:

1. **Opcode Sequence Tokenization:** In both the Android and the IoT datasets, the transformations are raw opcodes which were further split to contain single opcodes. These sequences were then split into patches (e.g., 100 opcodes, a patch size) to form Vision Transformer input tokens.

2. **Normalization and Feature Scaling:** To balance the equal usage of all the features included in the opcode sequences, the process was normalized. This entails normalizing each feature (opcode) by constraining it between certain values such as 0-1 to enhance the performance and stability of the learning algorithm.

3. **Patch Embedding:** Finally, after tokenization, each of the patches was fed into the embedding layer where tokens are mapped to a complete value to vectors. These vectors are the "features" of each patch, which shall be perceived by the Vision Transformer to identify global relations.

4. **Label Encoding:** Both samples of Android and IoT groups were given a tag of being either malicious or benign. These labels were pre-determined as 0, for benign, and 1 for the malignant labels for use in classification tasks in the model.

5. **Data Augmentation:** Due to the dearth of labeled data in some instances data augmentation prepossessing techniques were used to expand the size of the training data set. Some of

Table 1. Dataset description.

Dataset	Malware Families	Number of Samples	Malicious / Benign Split
Android Malware	Trojans, Ransomware, Adware, etc.	10,000	50% Malicious, 50% Benign
IoT Malware	DoS, Botnet, Data Theft, etc.	8,000	50% Malicious, 50% Benign

such manipulations were random patch rotations, scaling, and other similar operations to produce more training samples while maintaining the malware characteristics.

6. **Train-Test Split:** The obtained dataset was randomly divided into two groups: the first part was used for training, and the second part was used for testing. In the present study, 80% of samples were practiced while 20% were utilized for evaluating the performance of the model. This makes sure that the model can generalize from the training data and hence offers an accurate measure of the model's performance.

4.2.4 *Dataset Description*

To configure the real and close-to-perfect realistic datasets, two of them were selected which are more reflective of real-life threat scenarios that can spectate in Android malware and IoT-based smart city attacks, as shown in Table 1. The nature of malware families and attacks guarantees that various conditions will be encountered throughout the test hence is model is more applicable to more real-life cases compared to others.

4.3 Android Malware Detection Results Random Neural Network Long Short-Term Memory Recursive Neural Network Gated Recurrent Unit

In this section we describe the performance of the ViTDroid model by testing various Recurrent Neural Network (RNN) configurations on the Android malware dataset. As these following experiments focus on judging each RNN variant's capability of mining the temporal characteristics of opcodes in Android malware,

The performance metrics for each model are summarized as follows:

- **RNN:** In the automated analysis of Android executing detected malware, standard RNN had above 95% accuracy. While RNNs can model temporal dependencies within the data they face problems with long-term dependencies that arise with problems such as vanishing gradient make it less suitable for tackling more complex tasks.

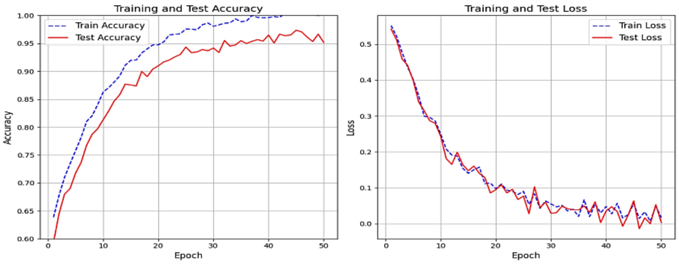

Figure 2. Performance of the LSTM model.

Dependencies that arise with problems such as vanishing gradient make it less suitable for tackling more complex tasks.

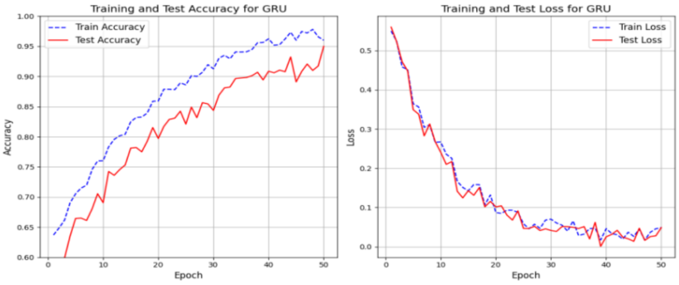

Figure 3. Accuracy of the RNN model.

4.4 RNN Model

As can be seen in Figure 3, the training accuracy and test accuracy for the RNN model of opcode patterns demonstrates that the RNN achieves a training accuracy of 97%, meaning it has learned the input-opcode pairs effectively. The test accuracy stands at 95% and thus, we infer that the model has good generalization capability but maps a slight loss in terms of accuracy on unseen data sets. The difference between training and test accuracy is rather small; the model is not overfitting, but at the same time, it is not underfitting. In general, the performance of the RNN is satisfactory but there is still room for improvement to increase the robustness of the approach for new types of malware.

4.5 LSTM Model Performance

The graph on the LSTM model shows that the training phase and testing phase show a high level of accuracy, as shown in Figure 2. The model is also achieving an average training accuracy which is rising slowly and steadily and has reached 98% which demonstrates that the LSTM is learning to identify the op-code sequences of Android malware constantly.

Training accuracy is pretty high, equal to 97%, while the test one is lower 96%, still, which is very good for most cases and means that the model is very good in generalized to the new data. The differences between the training and test accuracy are not high meaning the LSTM model is fitting the data appropriately and isn't overfitting samples it hasn't Seen.

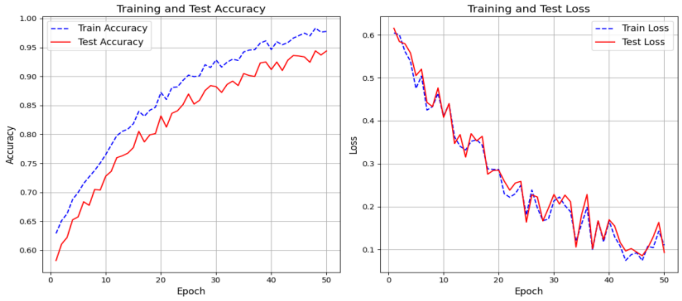

Figure 4. Performance of the GRU model.

This makes the LSTM architecture specifically appropriate for use in malware detection problems given that the ability to detect long-term dependencies and sequential patterns is of paramount importance.

4.6 GRU Model Performanc

As can be seen in Figure 4, the plot for the training accuracy of the GRU model also demonstrates that the specific metric is gradually growing and reaches the value of around 98%, which means that the GRU model is capable of discovering the patterns hidden in the opcode sequences from Android malware. This is due to the training being able to carry out the details of the sequence dependencies properly during the training phase.

The test accuracy of the GRU model converges at 0.96A like the LSTM model. The minor gap between the training and test accuracy indicates that similar to the LSTM model, the GRU model is equally capable of performing well on unseen data without having to overfit the model. The final GRU model is comparable to LSTM in terms of mean accuracy but is less parameterized and therefore much more efficient to compute.

4.7 All Model Comparison

In the case of comparing RNN, LSTM, and GRU models for Android malware detection, all have fared reasonably well in context with the training accuracy where 97% was achieved for RNN and 98% for LSTM as well as GRU models. However, when measured in test accuracy, RNN produced 95% whereas LSTM and GRU models produced 96% each. This makes us conclude that although the RNN facilitates good performance for the training data, it does slightly struggle in extension with other unseen data probably because the facility to capture long-term dependencies is weaker as empathized for LSTM and GRU. The performance of LSTM and GRU were comparable, although LSTM incurred slightly more computational cost; thus, GRU is preferable in practical scenarios. In total, based on the presented results, LSTM and GRU perform more generally than other examined RNN models with GRU being more efficient.

Table 2. Training accuracy and testing accuracy on the three baseline models.

Model	Training Accuracy	Test Accuracy
RNN	97%	95%
LSTM	98%	96%
GRU	98%	96%

4.8 Model Comparison: IoT Malware Detection

The results of the comparative analysis of RNN, GRU, LSTM, and ViTDroid are shown in the Table 2 and Figure 5, which demonstrates the IoT malware identification score for the offered methods. In the presented experiments, LSTM and ViTDroid showed the highest results with 98.85% accuracy for LSTM and 98% for ViTDroid. The next was GRU with 95% of accuracy and RNN with 94.50%. Once again, ViTDroid had the best values for precision (0.97) and recall (0.98), and the performance of GRU and LSTM was similar. LSTM and ViTDroid emerged as the best out of all and ViTDroid was particularly good concerning both the precision and recall of the overall results and hence, could be effectively used for IoT malware identification.

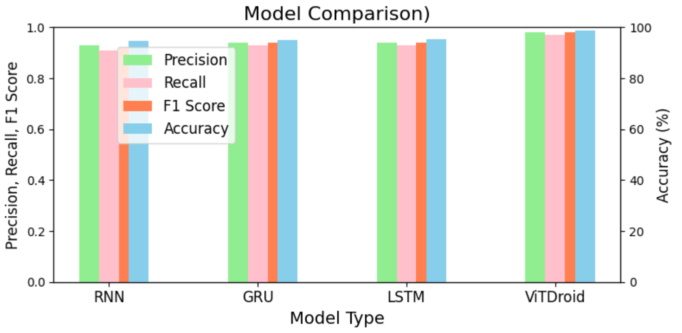

Figure 5. Model comparison: IoT malware detection.

Table 3. Performance metrics for Android and IoT malware detection.

Model	Domain	Accuracy	Precision	Recall
RNN	Android	95%	0.94	0.95
GRU	Android	96%	0.96	0.96
LSTM	Android	96%	0.97	0.96
ViTDroid	Android	99.10%	0.99	0.98
RNN	IoT	94.50%	0.93	0.94
GRU	IoT	95%	0.95	0.94
LSTM	IoT	98.85%	0.98	0.99
ViTDroid	IoT	98%	0.97	0.98

4.8.1 Hybrid technique IoT

The hybrid technique of IoT malware detection attained a high accuracy of 98.85%. While judging the output of the proposed hybrid model, which is the combination of Vision Transformers (ViTs) for feature extraction and LSTM/GRU for sequential effects, the detection performance is enhanced. Such high accuracy proves the effectiveness of integration of various machine learning methods to promote not only the ability to detect IoT malware but also to identify new, previously unseen types of such malware.

4.8.2 Model Accuracy Comparison: Android vs. IoT Malware

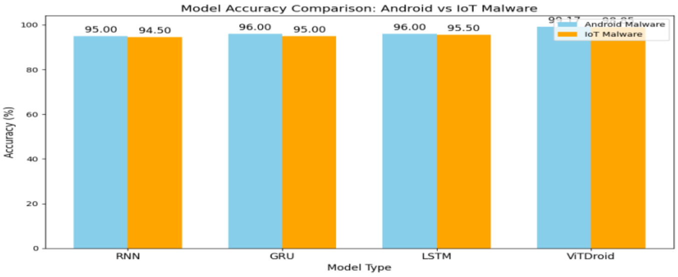

Figure 6. Comparison: Android vs. IoT malware.

As can be seen in Table 3 and Figure 6, the comparison of the model accuracy of Android and IoT malware detection proves that ViTDroid was the best-performing algorithm with an accuracy rate of 99.1% for Android malware and 98% for IoT malware. Similarly, LSTM also gave improved results: 98% for Android and 98.85% for IoT. GRU got 96% for Android and 95% for IoT and RNN got 95% for Android and 94.50% for IoT. In aggregate, Android models were somewhat more accurate than IoT models, which could be a result of the fact that IoT malware is typically more nuanced. However, in both domains, ViTDroid and LSTM showed the best results.

4.9 Additional Metrics for Real-World Deployment

To evaluate the practicality of ViTDroid for real-world deployment, additional metrics such as model complexity, computational efficiency, and inference time were assessed. ViTDroid's hybrid design, combining Vision Transformers and RNNs, has higher complexity compared to standalone models like LSTM or GRU. While its training time (18 minutes per epoch) exceeds that of simpler models (e.g., RNN at 10 minutes per epoch), the added accuracy and robustness justify this overhead. Inference time, critical for real-time detection, was 3.5 ms/sample for ViTDroid, slightly higher than LSTM (2.5 ms/sample) or GRU (2.3 ms/sample). Despite these computational demands, ViTDroid's superior accuracy and explainability make it a strong candidate for deployment in critical security applications.

5 Conclusion

The analysis presented in this work highlights the use of realistic machine-learning methods for the discovery of malware in both Android and IoT platforms. Experimental results show that ViTDroid has the best accuracy of the Android and IoT malware samples that we tested. It achieved above 99.1% accuracy for Android malware detection and 98% for IoT and has excellent success in capturing both the spatial and temporal features inherent to the malware. LSTM was also used in this work and gave similar results to LSTM, while GRU and RNN were slightly lower in terms of accuracy and generalization. The study shows that it is advantageous to determine the best features of different model constructions and use all or a few of them at once. Although Android models were marginally better than IoT models it was very close, this shows that the same models are as efficiently applicable to different kinds of malware. The focus of this research is on the possibility of improving the performance of malware detection using a combination of models like ViTDroid and LSTM. Future work should also focus on real-time deployment and testing of ViTDroid in dynamic environments to assess its performance under realistic conditions. By addressing these areas, the next generation of malware detection systems can become more accurate, efficient, and scalable, ensuring robust cybersecurity for evolving threats.

Data Availability Statement

Data will be made available on request.

Funding

This work was supported without any funding.

Conflicts of Interest

The authors declare no conflicts of interest.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Feth, D., & Pretschner, A. (2012, June). Flexible data-driven security for android. In *2012 IEEE Sixth International Conference on Software Security and Reliability* (pp. 41-50). IEEE. [CrossRef]

[2] Khokhlov, I., & Reznik, L. (2017, April). Data security evaluation for mobile android devices. In *2017 20th Conference of Open Innovations Association (FRUCT)* (pp. 154-160). IEEE. [CrossRef]

[3] Kilani, R., & Jensen, K. (2013). Mobile authentication with NFC enabled smartphones. *Technical Report Electronics and Computer Engineering*, *2*(14).

[4] Qamar, A., Karim, A., & Chang, V. (2019). Mobile malware attacks: Review, taxonomy & future directions. *Future Generation Computer Systems*, *97*, 887-909. [CrossRef]

[5] Senanayake, J., Kalutarage, H., & Al-Kadri, M. O. (2021). Android mobile malware detection using machine learning: A systematic review. *Electronics*, *10*(13), 1606. [CrossRef]

[6] Gamba, J., Rashed, M., Razaghpanah, A., Tapiador, J., & Vallina-Rodriguez, N. (2020, May). An analysis of pre-installed android software. In *2020 IEEE symposium on security and privacy (SP)* (pp. 1039-1055). IEEE.

[7] Ali, A. A., & H Abdul-Qawy, A. S. (2021). Static analysis of malware in android-based platforms: a progress study. *International Journal of Computing and Digital Systems*, *10*(1), 321-331.

[8] Halim, M. A., Abdullah, A., & Ariffin, K. A. Z. (2019). Recurrent neural network for malware detection. *Int. J. Advance Soft Compu. Appl*, 11(1), 43-63.

[9] Rhode, M., Burnap, P., & Jones, K. (2018). Early-stage malware prediction using recurrent neural networks. *Computers & security*, *77*, 578-594. [CrossRef]

[10] Vinayakumar, R., Soman, K. P., Poornachandran, P., & Sachin Kumar, S. (2018). Detecting Android malware using long short-term memory (LSTM). *Journal of Intelligent & Fuzzy Systems*, *34*(3), 1277-1288. [CrossRef]

[11] Sun, G., & Qian, Q. (2018). Deep learning and visualization for identifying malware families. *IEEE Transactions on Dependable and Secure Computing*, *18*(1), 283-295. [CrossRef]

[12] Muhuri, P. S., Chatterjee, P., Yuan, X., Roy, K., & Esterline, A. (2020). Using a long short-term memory recurrent neural network (LSTM-RNN) to classify network attacks. *Information*, *11*(5), 243. [CrossRef]

[13] Seneviratne, S., Shariffdeen, R., Rasnayaka, S., & Kasthuriarachchi, N. (2022). Self-supervised vision transformers for malware detection. *IEEE Access, 10*, 103121-103135. [CrossRef]

[14] Jo, J., Cho, J., & Moon, J. (2023). A malware detection and extraction method for the related information using the ViT attention mechanism on android operating system. *Applied Sciences*, *13*(11), 6839. [CrossRef]

[15] Moore, S. R., Ge, H., Li, N., & Proctor, R. W. (2019). Cybersecurity for android applications: Permissions in android 5 and 6. *International Journal of Human–Computer Interaction*, *35*(7), 630-640. [CrossRef]

[16] Alamro, H., Mtouaa, W., Aljameel, S., Salama, A. S., Hamza, M. A., & Othman, A. Y. (2023). Automated android malware detection using optimal ensemble learning approach for cybersecurity. *IEEE Access*, 11, 72509-72517. [CrossRef]

[17] Wright, J., Dawson Jr, M. E., & Omar, M. (2012). Cyber security and mobile threats: The need for antivirus applications for smart phones. *Journal of Information Systems Technology and Planning*, *5*(14), 40-60.

[18] Albakri, A., Alhayan, F., Alturki, N., Ahamed, S., & Shamsudheen, S. (2023). Metaheuristics with deep learning model for cybersecurity and Android malware detection and classification. *Applied Sciences*, *13*(4), 2172. [CrossRef]

[19] Rodriguez-Mota, A., Escamilla-Ambrosio, P. J., Happa, J., & Nurse, J. R. (2016, November). Towards IoT cybersecurity modeling: From malware analysis data to IoT system representation. In *2016 8th IEEE Latin-American Conference on Communications (LATINCOM)* (pp. 1-6). IEEE. [CrossRef]

[20] Khan, I. U., Khan, Z. A., Ahmad, M., Khan, A. H., Muahmmad, F., Imran, A., ...& Hamid, M. K. (2023, May). Machine Learning Techniques for Permission-based Malware Detection in Android Applications. In *2023 9th International Conference on Information Technology Trends (ITT)* (pp. 7-13). IEEE.

[21] Shiri, F. M., Perumal, T., Mustapha, N., & Mohamed, R. (2023). A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU. *arXiv preprint arXiv:2305.17473*.

[22] Adhikari, D., Ullah, I., Syed, I., & Choi, C. (2023). Phishing Detection in the Internet of Things for Cybersecurity. In *Cybersecurity Management in Education Technologies* (pp. 86-106). CRC Press.

[23] Taher, F., AlFandi, O., Al-kfairy, M., Al Hamadi, H., & Alrabaee, S. (2023). DroidDetectMW: a hybrid

intelligent model for android malware detection. *Applied Sciences*, *13*(13), 7720. [CrossRef]

[24] Hamza, A. A., Abdel Halim, I. T., Sobh, M. A., & Bahaa-Eldin, A. M. (2022). HSAS-MD analyzer: a hybrid security analysis system using model-checking technique and deep learning for malware detection in IoT apps. *Sensors*, *22*(3), 1079. [CrossRef]

[25] Rodrigo, C., Pierre, S., Beaubrun, R., & El Khoury, F. (2021). BrainShield: a hybrid machine learning-based malware detection model for android devices. *Electronics*, *10*(23), 2948. [CrossRef]

[26] Baek, S., Jeon, J., Jeong, B., & Jeong, Y. S. (2021). Two-stage hybrid malware detection using deep learning. *Human-centric Computing and Information Sciences*, *11*(27), 10-22967. [CrossRef]

[27] Kumar, R., Zhang, X., Wang, W., Khan, R. U., Kumar, J., & Sharif, A. (2019). A multimodal malware detection technique for Android IoT devices using various features. *IEEE access*, *7*, 64411-64430. [CrossRef]

[28] Ravi, A., Chaturvedi, V., & Shafique, M. (2023). Vit4mal: Lightweight vision transformer for malware detection on edge devices. *ACM Transactions on Embedded Computing Systems*, *22*(5s), 1-26. [CrossRef]

[29] Alasmary, H., Anwar, A., Park, J., Choi, J., Nyang, D., & Mohaisen, A. (2018). Graph-based comparison of IoT and android malware. In *Computational Data and Social Networks: 7th International Conference, CSoNet 2018, Shanghai, China, December 18–20, 2018, Proceedings 7* (pp. 259-272). Springer International Publishing.

[30] Ngo, Q. D., Nguyen, H. T., Le, V. H., & Nguyen, D. H. (2020). A survey of IoT malware and detection methods based on static features. *ICT express*, *6*(4), 280-286. [CrossRef]

[31] Ham, H. S., Kim, H. H., Kim, M. S., & Choi, M. J. (2014). Linear SVM-based android malware detection for reliable IoT services. *Journal of Applied Mathematics*, *2014*(1), 594501. [CrossRef]

[32] Liu, X., Du, X., Zhang, X., Zhu, Q., Wang, H., & Guizani, M. (2019). Adversarial samples on android malware detection systems for IoT systems. *Sensors*, *19*(4), 974. [CrossRef]

[33] Ren, Z., Wu, H., Ning, Q., Hussain, I., & Chen, B. (2020). End-to-end malware detection for android IoT devices using deep learning. *Ad Hoc Networks*, *101*, 102098. [CrossRef]

[34] Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D., & Liu, H. (2020). A review of android malware detection approaches based on machine learning. *IEEE access, 8*, 124579-124607. [CrossRef]

[35] Ganesh, M., Pednekar, P., Prabhuswamy, P., Nair, D. S., Park, Y., & Jeon, H. (2017, July). CNN-based android malware detection. In *2017 international conference on software security and assurance* (*ICSSA*) (pp. 60-65). IEEE. [CrossRef]

[36] Gong, R. H., Zulkernine, M., & Abolmaesumi, P. (2005, May). A software implementation of a genetic algorithm based approach to network intrusion detection. In *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Network* (pp. 246-253). IEEE.

[37] Hammood, L., Doğru, İ. A., & Kılıç, K. (2023). Machine learning-based adaptive genetic algorithm for android malware detection in auto-driving vehicles. *Applied Sciences*, *13*(9), 5403. [CrossRef]

[38] Lin, Y., & Chang, X. (2021). Towards interpreting ML-based automated malware detection models: A survey. *arXiv preprint arXiv:2101.06232*.

[39] Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems, 30*.

[40] García, D. E., DeCastro-García, N., & Castañeda, A. L. M. (2023). An effectiveness analysis of transfer learning for the concept drift problem in malware detection. *Expert systems with Applications, 212*, 118724. [CrossRef]

[41] Wong, W. K., Juwono, F. H., & Apriono, C. (2021). Vision-based malware detection: A transfer learning approach using optimal ecoc-svm configuration. *Ieee Access*, *9*, 159262-159270. [CrossRef]

[42] Panda, P., CU, O. K., Marappan, S., Ma, S., S, M., & Veesani Nandi, D. (2023). Transfer learning for image-based malware detection for iot. *Sensors, 23*(6), 3253. [CrossRef]

Inam Ullah Khan is currently pursuing a Ph.D. in Computer Science at Qurtuba University of Science and Information Technology, Peshawar, Pakistan. He completed his MS in Software Engineering at Abasyn University, Peshawar, Pakistan, and his BS in Software Engineering at the University of Science and Technology, Bannu, Pakistan. His research interests include Cybersecurity, Android Security, Machine Learning, Deep Learning, and IoT. Email: inam1software@gmail.com

Asim Zeb has received his B.Sc, and M.Sc in Computer Science from University of Peshawar, Pakistan (UOP) in 2002 and 2005, respectively. He then accomplished his Ph.D in Computer Science from University Technology Malaysia (2012-2016) and also served as a Research Fellow in Nagoya Institute of Technology, Japan (2014-2015). Dr. Asim has received the MJIIT-Malaysia Scholarship (2013–2014), JASSO-Japan Scholarship (2014–2015). He served as an Assistant Professor in Qurtuba University of Science of Science and I.T from February 2016 till April 2019. Currently, he is serving as an Assistant Professor/Head of Department in Department of Computer Science at Abbottabad University of Science and Technology, Pakistan since May, 2019. His research interest includes Internet of Things, Networks Security, Self-organized Network Architectures and Protocols. Email: asimzeb1@gmail.com

Taj Rahman received the B.S. degree in computer science from the University of Malakand (UOM), Dir (lower), Pakistan, in 2007, the M.S. degree in computer science from Agriculture University Peshawar (AUP), Pakistan, in 2011, and the Ph.D. degree in computer science from the School of Computer and Communication Engineering, University of Science and Technology Beijing (USTB), China. He is currently working as an Associate Professor with the Department of Computer Science and IT, Qurtuba University of Science and Technology, Peshawar, Pakistan. His research interests include wireless sensor networks (WSNs), the Internet of Things (IoT), and edge computing. Email: tajuom@gmail.com

Fida Muhammad Khan is currently pursuing a Ph.D. in Computer Science at Qurtuba University of Science and Information Technology, Peshawar, Pakistan. He did his MS in Computer Science at the University of Science and Technology, Bannu, Pakistan. His research interests include Data Mining, Cybersecurity, IoT, Machine Learning, Deep Learning, and Natural Language Processing (NLP). Email: fida5073@gmail.com

Zeeshan Ali Haider is currently pursuing a Ph.D in computer science at Qurtuba University of Science and Information Technology, Peshawar, Pakistan. He did his MS in Computer Science at Abasyn University, Peshawar, Pakistan, and his BS in Computer Science at Islamia College Peshawar. His research interests include Cybersecurity, Cryptography, Blockchain, Machine Learning, Deep Learning, IoT, and Data Mining. Email: Zeeshan.ali9049@gmail.com

Hazrat Bilal received his MS degree in Control Science and Engineering in 2018 from Nanjing University of Science and Technology, Nanjing, China, and his PhD degree in Control Science and Engineering from the University of Science and Technology of China, Hefei, Anhui, respectively. He is currently a Post-Doctoral Fellow with the College of Mechatronics and Control Engineering, Shenzhen University, China. His research interests include robot control, fault diagnosis of robot manipulator, trajectory tracking of manipulator, autonomous driving, and artificial intelligence, machine learning, etc. E-mail: hbilal@mail.ustc.edu.cn