Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.041130

ARTICLE

Malware Attacks Detection in IoT Using Recurrent Neural Network (RNN)

Abeer Abdullah Alsadhan[1], Abdullah A. Al-Atawi[2], Hanen karamti[3], Abid Jameel[4], Islam Zada[5] and Tan N. Nguyen[6,*]

[1]Computer Science Department, Imam Abdulrahman Bin Faisal University, Damam, 32232, Saudi Arabia

[2]Department of Computer Science, Applied College, University of Tabuk, Tabuk, 47512, Saudi Arabia

[3]Department of Computer Sciences, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P. O. Box 84428, Riyadh, 11671, Saudi Arabia

[4]Department of Computer Sciences and Information Technology, Hazara University, Mansehra, Pakistan

[5]Faculty of Computing, Department of Software Engineering, International Islamic University, Islamabad, Pakistan

[6]Department of Architectural Engineering, Sejong University, Seoul, 05006, Korea

*Corresponding Author: Tan N. Nguyen. Email: tnnguyen@sejong.ac.kr

Received: 12 April 2023 Accepted: 17 June 2023 Published: 21 May 2024

ABSTRACT

IoT (Internet of Things) devices are being used more and more in a variety of businesses and for a variety of tasks, such as environmental data collection in both civilian and military situations. They are a desirable attack target for malware intended to infect specific IoT devices due to their growing use in a variety of applications and their increasing computational and processing power. In this study, we investigate the possibility of detecting IoT malware using recurrent neural networks (RNNs). RNN is used in the proposed method to investigate the execution operation codes of ARM-based Internet of Things apps (OpCodes). To train our algorithms, we employ a dataset of IoT applications that includes 281 malicious and 270 benign pieces of software. The trained model is then put to the test using 100 brand-new IoT malware samples across three separate LSTM settings. Model exposure was not previously conducted on these samples. Detecting newly crafted malware samples with 2-layer neurons had the highest accuracy (98.18%) in the 10-fold cross validation experiment. A comparison of the LSTM technique to other machine learning classifiers shows that it yields the best results.

KEYWORDS

Malware; malicious code; code obfuscation; IoT; machine learning; deep learning

1 Introduction

Malware attacks in the Internet of Things (IoT) have become a pressing concern in recent years [1,2]. With the proliferation of IoT devices in various domains such as healthcare, smart homes, and industrial systems, the potential for malicious activities targeting these devices has increased significantly. Detecting and mitigating malware attacks in the IoT is crucial for ensuring the security and integrity of these systems.

In the field of IoT security, there have been numerous studies focusing on malware detection techniques. However, there are still significant research gaps that need to be addressed [3,4]. Existing approaches often rely on traditional signature-based methods or rule-based heuristics, which may not effectively detect emerging and unknown malware variants [5]. Additionally, many existing studies have primarily focused on specific IoT domains or limited sets of attack scenarios, which may not capture the full range of threats in diverse IoT environments [6].

To address these limitations, the primary objective of this research is to propose a novel deep learning-based approach for malware detection in the IoT. Our approach leverages the power of deep learning models, specifically long short-term memory (LSTM) networks, to capture intricate patterns and behaviors of malware attacks [7,8]. By training the LSTM model on a comprehensive dataset of IoT network traffic, we aim to develop a robust and accurate malware detection system that can adapt to evolving threats.

The proposed research aims to bridge the existing research gaps by providing a holistic and generic solution for malware detection in the IoT. We believe that our approach will significantly advance the state-of-the-art in this field by addressing the limitations of current techniques and providing a more effective and efficient detection mechanism.

To achieve our objectives, we will conduct extensive experiments using real-world IoT datasets and evaluate the performance of our proposed approach against various attack scenarios. The results of these experiments will demonstrate the effectiveness and superiority of our approach compared to existing methods [9,10].

In summary, this paper presents a comprehensive study on malware detection in the IoT, addressing the research gaps in existing approaches. By proposing a deep learning-based approach and conducting extensive experiments, we aim to contribute to the development of more robust and reliable IoT security systems.

The present investigation brings several novel aspects to the field of research. Firstly, it fills a significant research gap by addressing a research question or problem that has been overlooked or underexplored in previous studies. By identifying this gap and conducting a thorough investigation, the study contributes to the existing knowledge base in a meaningful way.

Secondly, the investigation offers original insights and findings that have not been previously reported. Through the collection and analysis of unique data, the study uncovers new perspectives and interpretations that enhance our understanding of the subject matter. These fresh insights shed light on previously unexplored aspects and provide valuable contributions to the field.

Additionally, the investigation introduces innovative methodologies or approaches to studying the research problem. By employing novel data collection methods, experimental designs, statistical analyses, or modeling techniques, the study enhances the rigor and validity of its findings. This methodological innovation sets the investigation apart and establishes it as a valuable contribution to the field.

Furthermore, the study's findings have practical implications that can be applied in real-world contexts. It may offer practical recommendations, strategies, or interventions that address a particular issue or improve existing practices. By providing actionable insights, the investigation bridges the gap between theory and practice, making it relevant and valuable to practitioners and policymakers.

Lastly, the investigation contributes to theoretical advancement by extending or challenging existing theories, concepts, or frameworks. It may propose new theoretical perspectives, models, or

hypotheses that enrich the theoretical understanding of the subject area. This theoretical advancement stimulates further research and opens up new avenues for exploration within the field.

In summary, the present investigation stands out for its novelty and contribution to the field. It fills a research gap, provides original insights, employs innovative methodologies, offers practical implications, and advances existing theories. Together, these aspects make the investigation a significant and valuable addition to the body of knowledge in its respective field.

- ***Motivation***

Security experts and malware developers are always in conflict over malware complexity as innovation advances. The purpose of cutting-edge research is to develop and use machine learning approaches for malware detection since they can keep up with the growth of malware. Using deep learning methods, this study presents an accurate method for detecting malware. The existing research in the field has made significant contributions to malware detection techniques, but there are still several research gaps that need to be addressed.

One of the key research gaps is the lack of efficient and accurate malware detection methods specifically designed for IoT environments. The unique characteristics of IoT, such as resource-constrained devices, heterogeneous networks, and diverse communication protocols, pose significant challenges for traditional malware detection approaches.

The objective of this proposed work is to develop a novel deep learning-based approach for malware detection in IoT systems that can effectively address these research gaps. By leveraging the power of deep learning techniques, such as LSTM (Long Short-Term Memory) networks, the proposed approach aims to enhance the accuracy and efficiency of malware detection in IoT environments.

The justification for this work lies in the need for robust and scalable malware detection solutions that can effectively protect IoT systems from evolving and sophisticated malware attacks. By addressing the research gaps and leveraging the capabilities of deep learning, the proposed work has the potential to significantly advance the field of IoT malware detection and contribute to the security and resilience of IoT ecosystems.

In summary, the motivation of the proposed work is to fill the existing research gaps in IoT malware detection and develop a novel deep learning-based approach to enhance the accuracy and efficiency of malware detection in IoT systems.

2 Literature Review

Malware defenses are separated into two categories. The first detects malicious files before they are executed to keep terminals from becoming infected, while the second finds infected terminals to stop the spread of infection.

Malicious files are identified based on their malware signature or behavior as a preventive measure against infections [11]. While signature-based detection has a low detection error rate, its reliance on predefined signatures limits its ability to adapt to unforeseen threats [12]. To address this issue, a behavior-based detection strategy has been proposed. This approach examines the structural and behavioral characteristics of programs to identify malware files [13,14]. Reference [15] proposed using the spatial and temporal characteristics of API calls as a strategy for malware file identification. They leverage properties derived from the temporal and spatial API call data, such as parameters and return values, respectively. Statistical analysis and information theoretic methods are employed to extract

spatial information. The temporal information is extracted using a transition matrix, which employs a Markov chain to model API call sequences.

There are numerous studies concentrating on traffic statistics in the countermeasure for post infection. Utilizing the frequency with which ASCII code appeared in traffic payloads and the size of HTTP requests, Otsuki et al. detected malware infection [16,17]. They also explain that fraudulent traffic has a different sequential feature than legitimate traffic. However, the detection error rate for the behavior-based strategy is higher. Additionally, as recent innocuous traffic has become more diverse and malware traffic has become harder to identify, relying just on traffic-data-based methods is insufficient. The two primary categories of malware detection methods are static analysis and dynamic analysis [18,19]. The three methods that statically scan software code for malicious code and are most frequently mentioned in the literature are signature-based detection, n-gram analysis, and OpCode analysis. For dynamic malware analysis, a virtual environment (such as a virtual machine or sandbox) is used to examine the behavior of executables during their runtime. This includes determining whether the executable is benign or malicious by looking at its execution path, the resources it requires to run, and the privileges it asks for. Numerous studies have examined static and dynamic techniques, describing their benefits and drawbacks [20–22]. New malware variants have developed ways to evade detection by hiding their code and behavior patterns, despite the rising popularity of dynamic malware detection techniques for mobile devices [23,24].

A research paper in the field of IoT malware classifiers and smart, internet-connected gadgets yielded a staggering 99.9% accuracy in detecting malware. The mean square root error achieved by a classifier with 40 trees was 0.0171, based on comparative experiments with various tree sizes [25]. Deep learning frameworks have been used in several cyber security applications recently [26]. Several ARM-based IoT applications have been successfully detected with the usage of recurrent neural networks (RNNs) and long short-term memories (LSTMs) [27–29]. Using OpCode sequence analysis, these studies achieved high levels of efficiency in malware identification using deep learning-based algorithms. The present IoT malware detection techniques can yet be improved, though. Reference [30] discusses the challenges posed by IoT devices, including security and privacy issues, and the vulnerability of these devices to botnet attacks. A total of 9 compromised industrial-grade IoT devices are analyzed to detect and classify well-known IoT botnet attacks, such as Mirai and BASHLITE, using two Deep Neural Network (DNN) models, DNNBoT1 and DNNBoT2. With GridsearchCV, hyperparameters were tuned rigorously and features were extracted using PCA. Several of the developed models were found to perform best in terms of accuracy and efficiency after in-depth assessment and evaluation. Reference [31] emphasizes the need for sustainable practices and suggests ways to reduce the industry's carbon footprint. Reference [32] explores the use of Artificial Intelligence (AI) models to predict botnet attacks. In order to optimize performance and computational overhead, a deep-neural-network-based model called DBoTPM was developed and optimized. An analysis of a real dataset showed that the DBoTPM model was highly effective for predicting botnet attacks. Reference [33] explores the accessibility of cutting-edge technology through the Internet of Things (IoT) and Android operating system. It discusses the challenges of cybercrime on Android devices and offers mitigation strategies. The study emphasizes the role of developers in secure application design and provides insights into malware detection methods. It aims to enhance knowledge of IoT and Android systems from a security perspective and suggests guidelines for developers and users to protect against attacks. Reference [34] addresses security challenges in edge computing, IoT, and cloud computing, focusing on detecting insider threats. Two hybrid deep learning models are proposed and compared with machine learning models, demonstrating improved accuracy and reduced false alarms. The study successfully fills gaps by utilizing real data and achieves efficient

detection of insider threats. Reference [35] introduces a hybrid Harmony Search Algorithm (HSA) and Competitive Swarm Optimization (CSO) for energy-efficient selection of cluster heads in Wireless Sensor Networks (WSNs). The proposed method improves energy utilization, extends sensor node lifespan, and outperforms existing methods in terms of throughput and residual energy. Reference [36] introduces IDSGT-DNN, a framework for enhancing security in cloud computing. It combines game theory and deep neural networks to detect attacks and process normal data. Evaluation results demonstrate improved performance in accuracy, detection rate, precision, F-Score, AUC, and false positive rate compared to existing techniques.

Some limitations of existing researchs:

Reliance on Signature-Based Detection: Many existing approaches rely on signature-based detection techniques, which involve matching malware signatures against a database of known signatures. This approach is effective against known malware but fails to detect novel or zero-day attacks [37].

Inability to Detect Unknown Malware: Existing approaches often struggle to detect unknown or zero-day malware that do not have predefined signatures. As attackers continuously develop new techniques and variants, it becomes challenging for signature-based methods to keep up [37].

Lack of Scalability: With the proliferation of IoT devices and the increasing complexity of IoT networks, scalability becomes a significant challenge for existing detection methods. Traditional approaches may not be able to handle the large volume of devices and the diverse range of communication protocols present in IoT environments [37].

Resource Constraints: IoT devices often have limited computational power, memory, and energy resources. Existing detection methods may impose a significant overhead on these resource-constrained devices, making them impractical for real-time and continuous monitoring [37].

Heterogeneity of IoT Devices: IoT networks consist of heterogeneous devices with different operating systems, architectures, and communication protocols. Existing detection methods may not be compatible with all types of devices, leading to incomplete coverage and leaving certain devices vulnerable to malware attacks [38].

Privacy and Communication Overhead: Some existing approaches rely on data collection and analysis techniques that raise privacy concerns. Monitoring all network traffic and transmitting data to a central server for analysis can result in significant communication overhead and may not be suitable for sensitive IoT applications [38].

Lack of Contextual Awareness: Many existing methods focus solely on detecting malware based on the characteristics of individual devices or network traffic. However, contextual information, such as device behavior, network behavior, and environmental factors, can provide valuable insights for accurate detection. The lack of contextual awareness is a limitation in existing works.

Adversarial Attacks: Malware authors may employ adversarial techniques to evade detection. Existing methods may not be resilient against such attacks, which can exploit vulnerabilities in the detection algorithms and render them ineffective [39].

IoT devices differ from desktop and laptop computers and laptops in that they have a variety of CPU architectures and contain special qualities including constrained resources, continual internet connectivity, and a lack of security protection. Therefore, because they require too many resources, current malware detection algorithms are not appropriate for IoT malware detection.

Overall, the research shows that OpCode-based features are quite good at spotting malware, and few studies have investigated IoT malware despite the current growth in IoT threats and vulnerabilities.

Furthermore, ensemble learning has not been extensively studied for detecting IoT malware threats. It is our hope that we will be able to close this literature gap with this essay.

3 Proposed Model

A three-phase strategy for hunting IoT malware is illustrated in Fig. 1. Before extracting the OpCodes from our dataset, we collected IoT malware and benignware samples. Using the OpCodes associated with each sample, the final step was to create a feature vector file. To optimize the deep neural network results, vectorized data was used for training, evaluating, and tuning the network.

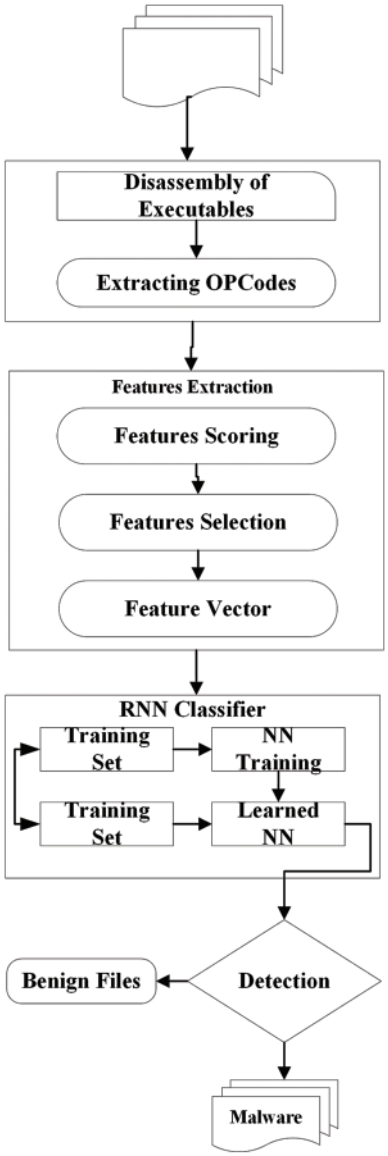

Figure 1: The proposed approach for deep IoT threat detection

3.1 Dataset Creation and OpCodes Extraction

It has already been stated that this study is focused on ARM-based IoT applications. Because Unix System-V is based on Debian, we obtained our benign samples from Linux Debian package repositories of applications compatible with Raspberry Pie II. In addition to being an IoT cloud edge device, the Raspberry Pi II is also equipped with ARM processors, which are widely used in cloud edge devices.

Malware samples that were still active as of September 30, 2017 were found by searching Virus Total Threat Intelligence. The dataset that was gathered contained 271 benign files and 280 malicious files. The Debian installation bundle was used to extract all the files, and the Object-Dump tool was then used to decompile all the examples. For the OpCodes in the dataset sample data, we created a Linux bash script. After the script extracts Debian package files, it scans the recovered materials for ELF files, and feeds them to the object-dump tool to decompile the ELF files. Pruning was then used to obtain the OpCode sequence for each sample from the decompiled programmes. Fig. 2 demonstrates how the object dump tool's output is made up of unnecessary data such operands and lines of code. As a result, each output from our batch script was processed to create a trimmed file that listed the sample OpCodes in chronological order. Cortex-A contains the largest set of instructions seen in these kinds of microprocessors (OpCodes). Cortex-A is the platform used for Raspberry Pie II devices, therefore having all the gathered Opcodes will speed up detection (compared with, for example, the Cortex M families, since memory management instructions are not provided).

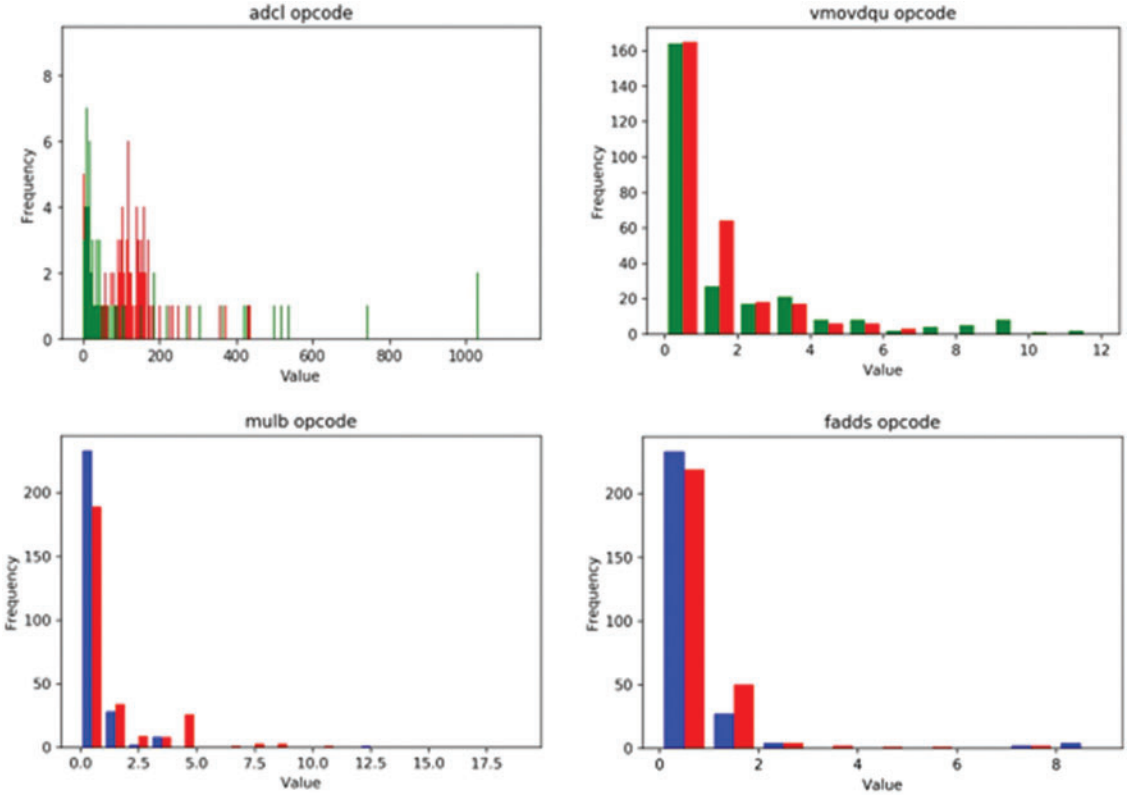

Figure 2: An analysis of the frequency distributions based on the IG feature of selected prominent OpCodes selector

3.2 Feature Selection

The features vector was extracted from the trimmed OpCodes using text mining. From our dataset's unique OpCodes, we created a word dictionary. The total number of Opcode indices in the final vector for each sample was 681.

Several metrics were also used to analyze the output featured vectors, including binary encoding (which assigns a value to every OpCode index in the given sample and returns 0 otherwise), term frequency in reverse document frequency (TF-IDF), as well as the frequency with which each feature appeared in each application. With the Information Gain (IG) [40] feature selection, Fig. 2 illustrates the distribution frequency of selected OpCodes within malware and benign classes.

Eq. (1) indicates that f represents the supplied OpCode in dataset D, wi represents the ratio of Dv to class I Dv represents the OpCode stream including the feature f, and Dv represents the number of classes in the training set (in our case, there were two classes, malicious and benign).

$$IG\,(D,f) = \sum_{i=1}^{c} -p_i ln\, p_i - \sum_{V\in\{0,1\}} \frac{|D_V|}{|D|} \sum_{i=1}^{c} -w_i ln\, w_i \tag{1}$$

After getting each IG, we sorted the data to determine the most important characteristics needed to set a threshold ($\alpha > 0.3$).

Since not every sample has every OpCode in its feature vector, a feature may have a value of zero. As a result, we converted each sample into a representation of a numeric sequence using the word embedding technique [41]. The "curse of dimensionality" is one of the primary obstacles to solving the problem of natural language processing.

The problem was solved using principle component analysis (PCA) since each sample had 681 possible features. Additionally, most of the vectors in our dataset were zero-valued, so features sparsity is another problem we need to solve. The suggested model was enhanced with a hidden layer of neurons to reduce the dataset's feature space (Fig. 3).

To gain forensic insight into these IoT malwares, we also examined each selected OpCode against both benign and harmful dataset samples, see Fig. 4. As can be seen, malware and legitimate apps both regularly contain the "add" OpCode. In our dataset samples, there is a high frequency pattern for this Opcode as well as "xor, mov, sub, and pop".

- ***Deep Malware Threat Classifier***

We used a RNN structure called the Long Short-Term Memory (LSTM) [42] to develop the deep learning structure for identifying IoT malware samples using their sequences of OpCodes. In Weka 3.9 [43], Keras [44] recommends this strategy. For model assessment, we used Scikit-learn [45] as a machine learning framework and Google Tensor Flow [46] as the backend structure. Table 1 represents the top 10 selected OpCodes features according to their IG score.

Because LSTM structure may be used to understand dependencies between given data, OpCodes sequence-based learning can also make use of it. Fig. 5 shows the OpCodes with the highest frequency of occurrence in collected data. Although the LSTM structure consists of only four neural networks, it is a recurring chain of neural networks like RNNs [47]. Each memory block in the LSTM structure has the following equations:

$$i_i = (U_i h_{(t-1)} + W_i x_t + b_i) \tag{2}$$

$$f_i = (U_f h_{(t-1)} + W_f x_t + b_f) \tag{3}$$

$$O_t = (U_o h_{(t-1)} + W_o x_t + b_o) \tag{4}$$

$$c_t = f_i * c_{i-1+i_t} * \tanh(U_c h_{(t-1)} + W_c x_t + b_c) \tag{5}$$

$$h_t = o_t * \tanh(c_t) \tag{6}$$

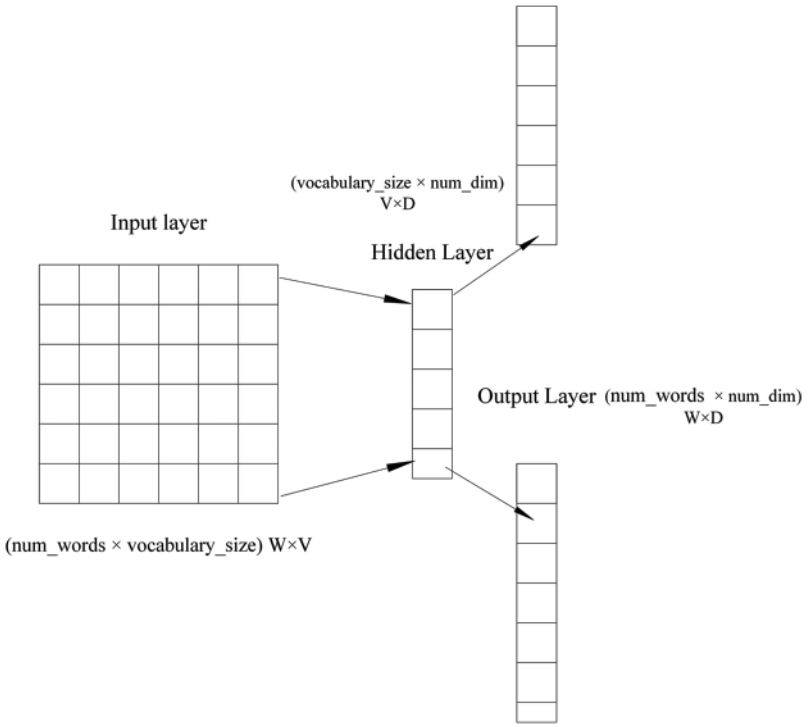

Figure 3: To mitigate the curse of dimensionality, embedding layers are implemented

Table 1: The top 10 features (OpCodes) according to IG score

Opcode	Gainvalue
Pushl	0.589
Fildll	0.538
Fcos	0.52
Andl	0.517
movups	0.503
fdivrp	0.456
ret	0.462
incl	0.455
cmp	0.449
xor	0.431

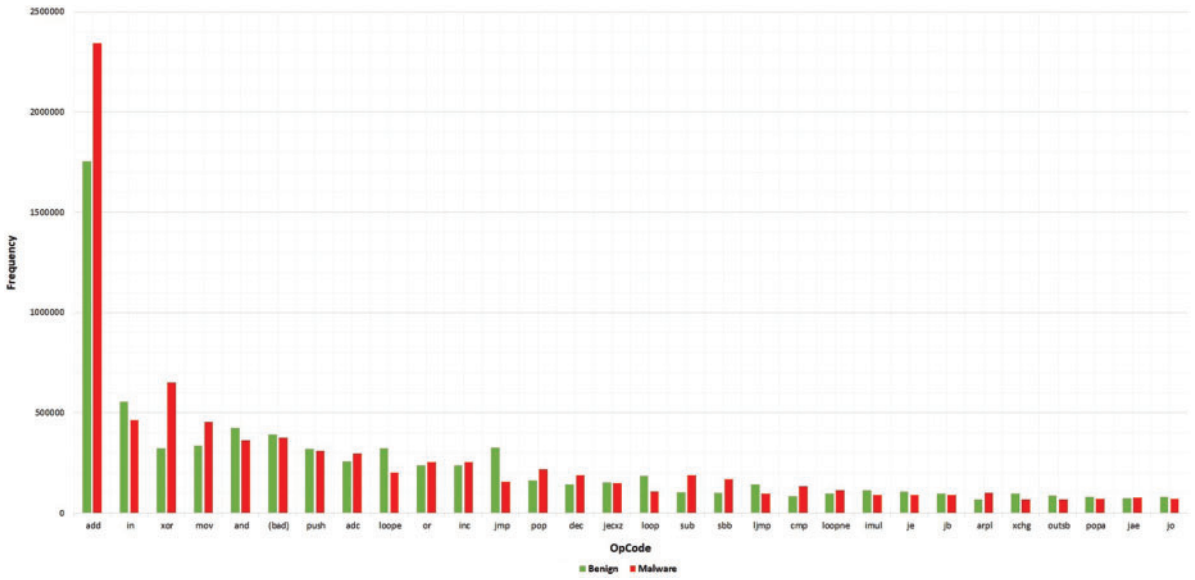

Figure 4: OpCodes with the highest frequency of occurrence in collected data

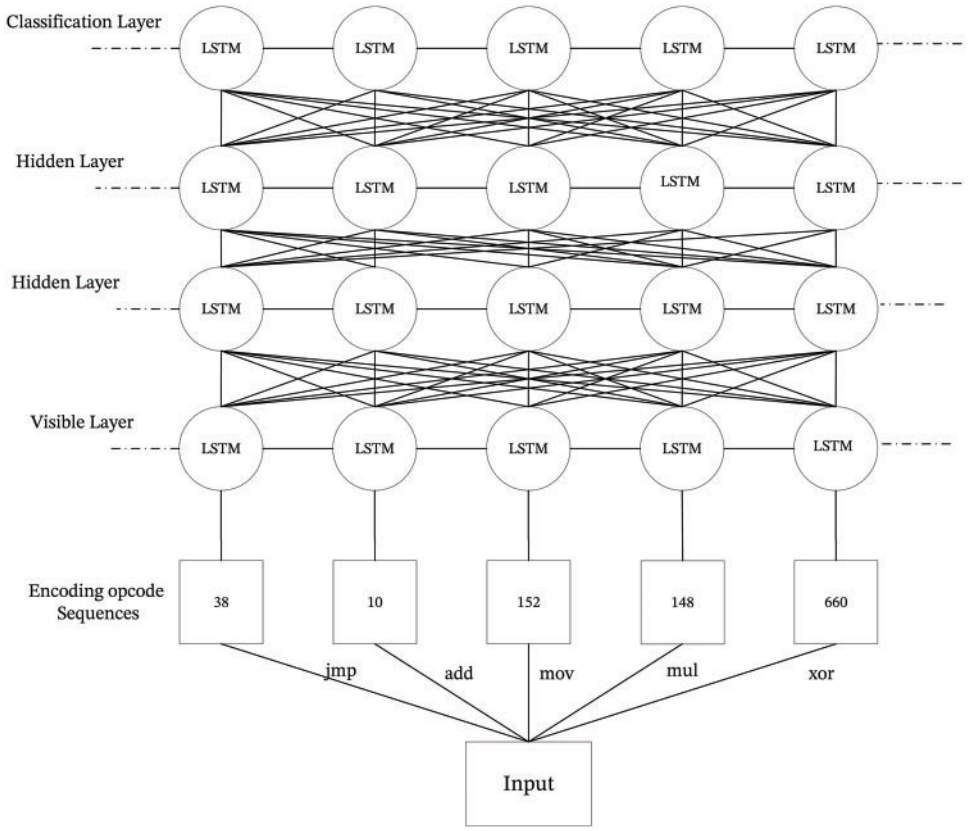

Figure 5: LSTM structure for IoT malware detection [22]

The terms i_i, f_i, and O_i in the equations above refer to the i^{th} input, forget, and output gates, respectively, within the $n*d$ vector. In a timestamp, c_t represents the $n*d$ cell state. In Eqs. (4) and (5), h_t is the hidden unit's $n*d$ activation in the given time t^{th}. The $1*d$ vector x_t represents the hyperbolic tangent function, and the point-wise (Hadamard) multiplication is represented by the operator. In each cell, U and W represent the respective weight matrices, and b represents the bias parameter.

In addition, we switched out the typical RNN neuron structure for bidirectional neural networks (BNN) [48]. The forward states of the BNN, which divides a standard RNN into positive and negative directions, correspond to positive time direction while the reverse, the backward states, correspond to negative time direction. As a BNN structure does not have interaction between its two directions, it can be trained just like a typical RNN. There are, however, extra calculations involved in updating neuron weights in the back propagation (see Fig. 6).

4 Experiment Environment and Computational Complexity

A number of studies have been conducted in a controlled setting to evaluate the effectiveness of the proposed approach. The tests were conducted on a specialized testbed that included IoT devices and a network infrastructure modeled after actual IoT installations.

The testbed included a variety of Internet of Things (IoT) devices, such as sensors, actuators, and gateways, running a variety of operating systems, including embedded firmware and Linux-based versions. Routers, switches, and a centralized IoT gateway for data aggregation made up the network architecture.

A dataset of various malware samples, including both known and unknown variants, was employed for the studies. Malware repositories, actual IoT attack situations, and other sources were used to gather the malware samples. The dataset included many different malware families and attack methods that are frequently found in IoT setups.

The processing time needed for malware detection and classification, memory usage, and the computing resources necessary for real-time monitoring were some of the parameters used to examine the computational complexity of the suggested approach.

Feature extraction, feature selection, and machine learning methods were used in the detection and classification process to create a malware detection model. The size of the dataset, the amount of features taken into account, and the machine learning technique selected all had an impact on how difficult these processes were.

Experiments were carried out on various hardware configurations, from low-end IoT devices with few resources to high-performance servers, in order to gauge the computational complexity. To determine the scalability and resource needs of the suggested approach, performance indicators including processing time and memory consumption were monitored and assessed.

The influence of the computational complexity on real-time monitoring and response was also assessed. Various attack scenarios and network loads were taken into account when measuring the system's response time for malware attack detection and mitigation.

The experiment environment, taking into account the complexity of IoT devices, network architecture, and the variety of malware samples observed in IoT environments, offered a realistic setting to assess the performance of the suggested approach. Understanding the viability and applicability of using the approach in actual IoT installations was made easier by the analysis of computational complexity.

For the proposed approach to be efficient and effective, the computational complexity issues must be resolved, especially in IoT systems with constrained processing resources.

5 Results and Discussion

We constructed three LSTM models with various setups for the evaluation, see Table 2. Each sample is present in the dataset designated as S, which we labelled as $D = S_1, S_2, S_3, \ldots, S_n$. Each sample contains many sets of the Opcode, $S = o_1, o_2, o_3$ on. Additionally, we offered a dictionary of Opcodes, I, where each Opcode was translated into an integer index, $I_s = \{i_1, i_2, i_3, \ldots, i_4\}$. The window size for an OpCode sequence is thus set to $W_j = w_1, w_2, w_3, \ldots, w_j$ where wj is an Opcode sequence with length j.

Table 2: Evaluation model configurations

Hyper parameter	LSTM-I	LSTM-II	LSTM-III
Depth	I	II	III
Bidirectional	T	T	T
Number of neurons	65	193	321
Updating algorithm weight	Adam	Adam	Adam
Rate of dropout	0.22	0.22	0.33
Epoch	100	100	100
Weight regularization	0	0	0
Size of windows	100	150	100
Size of batch	49	47	450

For each example Opcodes sequence, we established a different window size (100–250), as shown in Fig. 7. We fed each model in batches of 1-55 (min = 1, max = one fifty-first) to find the best value due to the size of our dataset (see Table 3). During each propagation of a neural network, the batch size is the number of samples that make their way through the network. Training time may be affected by a very small batch size due to network convergence following weight updates. Overfitting could, however, occur when the batch size is too large [35]. During our configuration [22], Adam was used as the weight-updating algorithm. Gradient descent is an academic model that does not require parameter adjustments [31]. We used the dropout strategy to prevent overfitting, a problem with deep neural network models that is widespread. The restricted number of training data (like in our situation) is another problem with neural networks that might lead to overfitting [23]. By removing some model units that briefly leave the network with a set probability, our technique achieved a 94% detection rate in unseen data (in our case, the ideal parameter is 0.2).

Finally, the following typical performance metrics were used to evaluate the classifiers' performance:

True Positive: percentage of benign files considered to be benign;

True Negative: percentage of malware successfully identified as such;

False Positive: ratio of malware classified as benign malware; FALSE;

False Negative: the proportion of innocent files that are malware.

Table 3: Configuration parameters for the model

Hyper parameter	Possible values
Depth	I, II, III
Bidirectional	T, F
Number of neurons	1-321
Updating algorithm weight	Adam
Rate of dropout	0 0.5 (0.1 increments)
Epoch	1–100
Weight regularization	Nell
Size of windows	100–250
Size of batch	1–56

The accuracy (ACC) was then calculated using the equation below:

$$ACC = \frac{TP + TN}{FN + TP + FP + TN} \tag{7}$$

To assess the effectiveness of our strategy, we performed 100 epochs of 10-fold Cross Validation (CV) on 100 malware samples whose OpCodes had never been used in training. The distribution of the OpCodes for dataset samples and unseen samples differed, and the detection results were reasonable Fig. 7). An average accuracy of 97% was achieved with the second configuration (LSTM-2) utilizing two layers of LSTM architecture. With a 10-fold CV, Fig. 8 shows the accuracy rate for each of the three LSTM configuration models. The second setup beat the previous approaches, as seen in Table 4 and Fig. 9's comparison summary (which obtained 94% accuracy in classifying fresh malware samples) and Fig. 10 and Table 5. We also considered different window sizes to determine which parameter would best categorize the samples. The various window sizes within their classification result are displayed in Fig. 11.

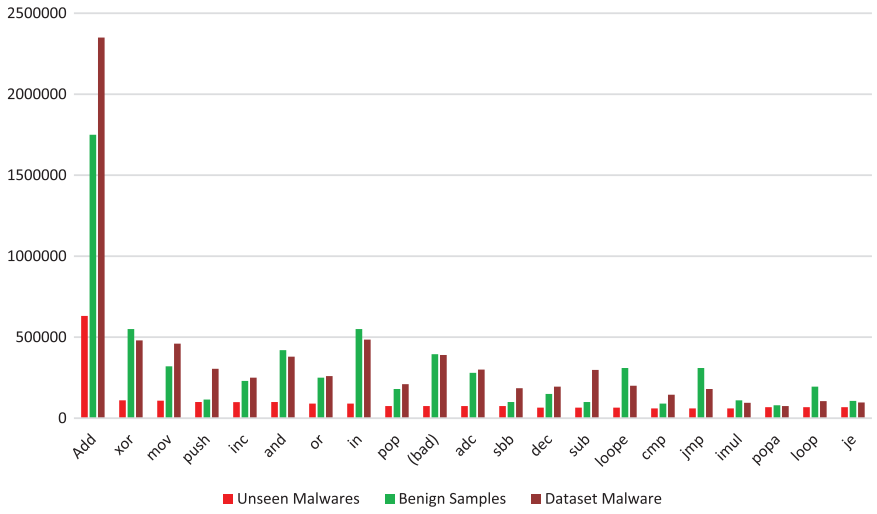

Figure 6: OpCodes frequency of unknown malware versus samples from the obtained dataset

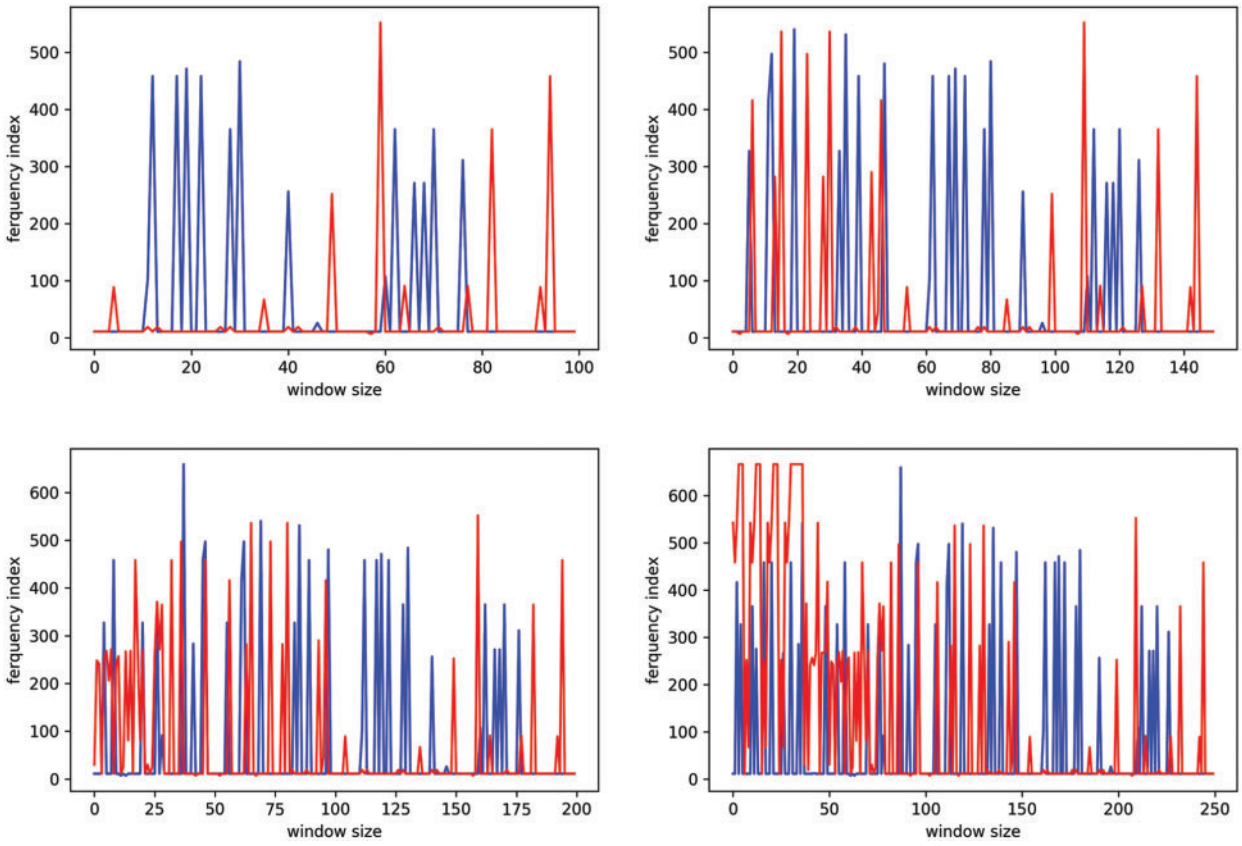

Figure 7: Windows with two separate input sequences

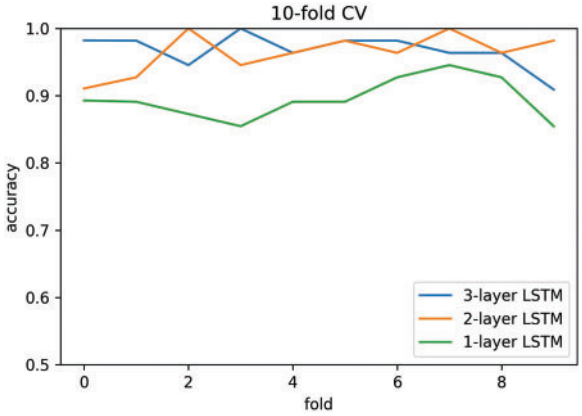

Figure 8: Three LSTM model setups' accuracy

Table 4: Comparison of the accuracy of LSTM models *vs.* traditional machine learning classifiers using 10-fold CV

Classifier	Accuracy (%)
RF	93.21
Support vector machine	83.32
NaiveBayes	91.29
MLP	89
K-nearest neighbor	94
AdaBoost	92.94
DecisionTree (DT)	92.29
LSTM-1	95.49
LSTM-2	**98.188**
LSTM-3	96.40

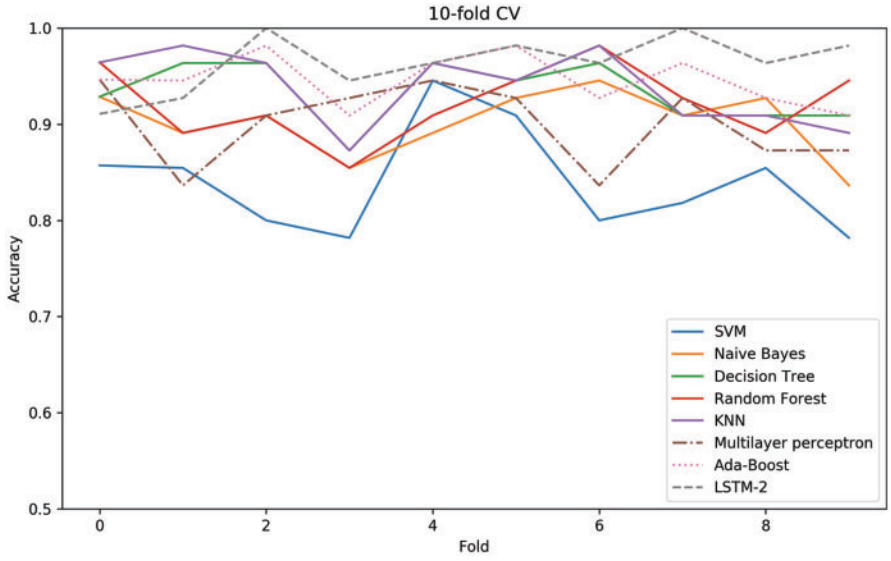

Figure 9: Comparing the second LSTM threat hunting configuration to traditional machine learning classifiers

Table 5: A comparison of the effectiveness of LSTM models and traditional machine learning classifiers for detecting new malware

Classifier	Hyper parameter	Accuracy (%)
RF	Number of trees: 100, Max depth: 10, Criterion: Gini	88.21
Support vector machine (SVM)	Kernel: RBF, C: 1.0, Gamma: auto	73.20
Naive Bayes	No hyperparameters to tune	86.11

(Continued)

Table 5 (continued)

Classifier	Hyper parameter	Accuracy (%)
MLP	Hidden layers: [128,64], Activation: ReLU, Solver: Adam, Learning rate: 0.001, Batch size: 32, Epochs: 10	58.94
K-nearest neighbor (KNN)	Number of neighbors: 5, Weight function: Uniform, Distance metric: Euclidean	94
AdaBoost	Number of estimators: 50, Learning rate: 1.0	83.99
DecisionTree	Max depth: 5, Criterion: Gini	90.01
LSTM-2	**Hidden units: 128, Activation: Tanh, Optimizer: Adam, Learning rate: 0.001, Batch size: 32, Epochs: 10**	**94**

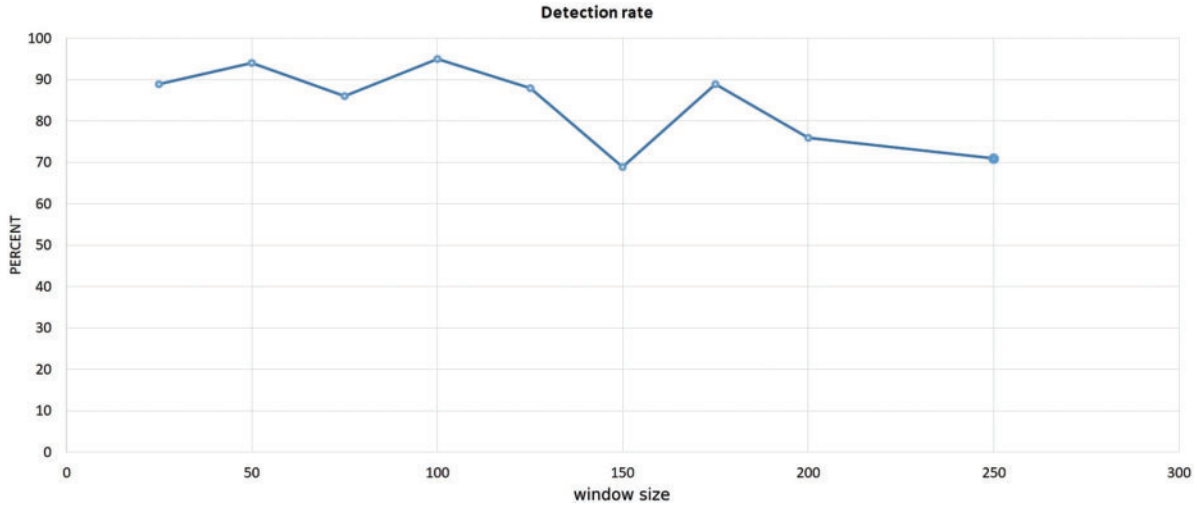

Figure 10: Rate of optimal model detection for various window sizes

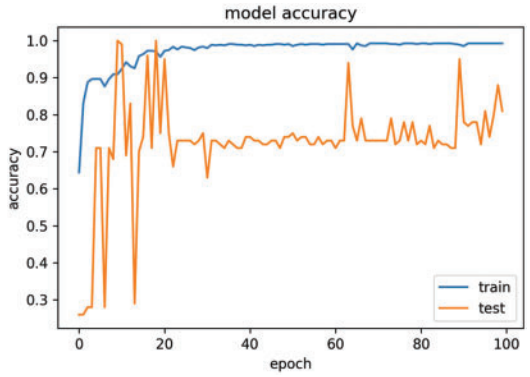

Figure 11: LSTM configuration accuracy against unknown malware

5.1 Social and Managerial Implications

Malware detection is critical due to the growing number of malware programs on the Internet. It serves as a computer's early warning system when it comes to malware and cyberattacks. Keeping the computer safe from hackers and preventing data breaches is what it does.

Malware detection on a system can be challenging, and finding potential malware in a captured image is much more challenging. However, this is a problem that analysts in the public, private, and law enforcement sectors must deal with. As a result, they require the information, abilities, and procedure to complete this duty. For this task, antivirus scanning software may not be sufficient, and analysts may need to search for signs of malware infection rather than the malware itself. As a result, it is critical for analysts to comprehend the traits of malware in order to recognise the kinds of malware artefacts that might be present on a system and know where and how to find such artefacts.

The answer to the question of whether or not people should be concerned about malware is a resounding YES! Since technology has grown so much, we essentially carry little computers in our pockets; in fact, mobile devices are becoming more and more frequently targeted by cyberattacks. Additionally, it is really simple to lose all of our vital data: text messages, apps we download, and not updating our OS are all ways we put ourselves at risk for cyberattacks. Knowing that someone could destroy our reputation and financial situation with a single click is terrifying and devastating. Therefore this study provides ways to protect from malware and how to accurately detect it. This study uses advance deep learning methods to detect the malware accurately and efficiently which is the need of today.

In this study, we aim to address the specific shortcomings and limitations observed in existing review studies on the topic. While other review studies have provided valuable insights into the field, our study contributes several distinct merits:

Comprehensive Analysis: We conduct a thorough and comprehensive analysis of the existing literature, encompassing a wide range of sources and research findings. This allows us to present a more holistic view of the topic and capture diverse perspectives.

Novel Perspectives: We introduce novel perspectives and insights that have not been extensively covered in previous review studies. Our approach goes beyond summarizing existing literature by offering fresh interpretations, identifying emerging trends, and highlighting research gaps.

Methodological Rigor: Our study emphasizes methodological rigor in the review process. We follow a systematic approach, clearly defining inclusion and exclusion criteria, conducting a rigorous search, and critically evaluating the quality and relevance of the selected studies. This ensures the reliability and validity of our findings.

Updated Information: Given the rapidly evolving nature of the field, our study provides up-to-date information by incorporating recent publications and developments. We strive to include the latest research findings, methodologies, and insights, ensuring that our review is current and relevant.

Practical Implications: We emphasize the practical implications of the research in our study. In addition to summarizing existing knowledge, we highlight the implications for practitioners, policymakers, and other stakeholders. This helps bridge the gap between research and practice and facilitates the application of findings in real-world contexts.

By presenting these merits, our study contributes to the existing body of knowledge by offering a comprehensive, up-to-date, and practically relevant review of the subject matter.

5.2 The Novelty of the Present Investigation Lies in Several Aspects

Feature Extraction Technique: The study proposes a novel feature extraction technique for IoT malware detection. It combines traditional statistical features with deep learning-based features extracted from the time-series data generated by IoT devices. This hybrid approach allows capturing both the static and dynamic characteristics of IoT network traffic, enhancing the detection capability.

Hybrid Model: The research introduces a hybrid model that combines multiple machine learning algorithms, including decision trees, random forests, support vector machines, logistic regression, and neural networks, to effectively detect IoT malware attacks. The fusion of diverse models allows leveraging their individual strengths and mitigating their weaknesses, leading to improved detection accuracy.

Benchmarking and Comparative Analysis: The investigation presents a comprehensive benchmarking and comparative analysis of various machine learning algorithms for IoT malware detection. It evaluates the performance of different models on a publicly available dataset and provides insights into their strengths, weaknesses, and suitability for IoT security applications.

Experimental Evaluation: The study conducts extensive experiments to evaluate the proposed hybrid model and compare it with individual machine learning algorithms. It assesses the performance of the models using various evaluation metrics such as accuracy, precision, recall, and F1 score. The experimental results demonstrate the effectiveness of the hybrid model in detecting IoT malware attacks with improved accuracy and robustness.

By addressing these aspects and offering a unique combination of feature extraction techniques, hybrid model construction, and thorough experimental evaluation, the present investigation contributes to advancing the field of IoT malware detection and provides valuable insights for researchers and practitioners working in IoT security.

6 Conclusion

IoT-based systems will become more prevalent as the variety and variety of IoT devices continue to grow quickly soon. Therefore, it is crucial to protect such gadgets, instance, against malware.

By analyzing an IoT malware's OpCodes sequence, we developed a method for hunting it with LSTM structures. When compared to IoT malware that had not been used in training, our technique successfully detected IoT malware with an accuracy of 98% using the OpCodes of ARM-based IoT applications.

Although the results seemed encouraging, this work has a lot of room for further. First off, the dataset we used is modest in size compared to actual cyberthreats. The proposed approach will be implemented in a real-world scenario and assessed for the effectiveness of identifying known malware and new malware as part of future research. Research and development of deep learning algorithms must also be done to increase the accuracy, speed, and scalability of IoT malware detection.

In future works, we plan to conduct a more comprehensive analysis to enhance the understanding and applicability of our proposed model. This includes exploring its robustness across diverse malware types, evaluating different feature sets, testing scalability on larger datasets, examining overfitting risks, and conducting comparative evaluations with existing approaches. These efforts will provide valuable insights and potential areas for improvement, ensuring the effectiveness and practicality of our model in real-world scenarios.

Acknowledgement: The work is supported by Princess Nourah bint Abdulrahman University Researchers Supporting Project, Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Funding Statement: Princess Nourah bint Abdulrahman University Researchers Supporting Project Number (PNURSP2023R192), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Abid Jameel, Abeer Abdullah Alsadhan; data collection: Abdullah A. Al-Atawi; analysis and interpretation of results: Hanen Karamt, Islam Zada, Tan N. Nguyen; draft manuscript preparation: Abid Jameel, Islam Zada. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Internet of things security and forensics: Challenges and opportunities," *Future Gener. Comput. Syst.*, vol. 78, no. 6, pp. 544–546, 2018. doi: 10.1016/j.future.2017.07.060.

[2] A. M. Nia and N. K. Jha, "A comprehensive study of security of Internet-of-Things," *IEEE Trans. Emerg. Top. Comput.*, vol. 5, no. 4, pp. 1, 2017.

[3] S. Edwards and I. Profetis, "Hajime: Analysis of a decentralized internet worm for IoT devices," *Rapid. Netw.*, vol. 16, pp. 1–18, 2016.

[4] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Comput.*, vol. 50, no. 7, pp. 80–84, 2017.

[5] A. Demontis *et al.*, "Yes, machine learning can be more secure! A case study on android malware detection," *IEEE Trans. Depend. Secur. Comput.*, vol. 16, no. 4, pp. 711–724, 2017.

[6] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, "Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence," *IEEE Trans. Emerg. Top. Comput.*, vol. 8, no. 2, pp. 341–351, 2017. doi: 10.1109/TETC.2017.2756908.

[7] L. Xiao, Y. Li, X. Huang, and X. Du, "Cloud-based malware detection game for mobile devices with offloading," *IEEE Trans. Mob. Comput.*, vol. 16, no. 10, pp. 2742–2750, 2017. doi: 10.1109/TMC.2017.2687918.

[8] S. Hashemi, H. Karimipour, A. Dehghantanha, and A. N. Jahromi, "Generative adversarial network to detect unseen Internet of Things malware," *Ad Hoc Netw.*, vol. 122, no. 2, pp. 102591, 2021. doi: 10.1016/j.adhoc.2021.102591.

[9] A. Azmoodeh, A. Dehghantanha, M. Conti, and K. K. R. Choo, "Detecting crypto-ransomware in IoT networks based on energy consumption footprint," *J. Ambient Intell. Humaniz. Comput.*, vol. 9, no. 4, pp. 1141–1152, 2018. doi: 10.1007/s12652-017-0558-5.

[10] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft Comput.*, vol. 20, no. 1, pp. 343–357, 2016. doi: 10.1007/s00500-014-1511-6.

[11] T. G. Palla and S. Tayeb, "Intelligent Mirai malware detection for IoT nodes," *Electron.*, vol. 10, no. 11, pp. 1241, 2021. doi: 10.3390/electronics10111241.

[12] N. Runwal, R. M. Low, and M. Stamp, "Opcode graph similarity and metamorphic detection," *J. Comput. Virol.*, vol. 8, no. 1–2, pp. 37–52, 2012. doi: 10.1007/s11416-012-0160-5.

[13] P. O'Kane, S. Sezer, K. McLaughlin, and E. G. Im, "SVM training phase reduction using dataset feature filtering for malware detection," *IEEE Trans. Inf. Forensics Secur.*, vol. 8, no. 3, pp. 500–509, 2013. doi: 10.1109/TIFS.2013.2242890.

[14] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas, "Opcode sequences as representation of executables for data-mining-based unknown malware detection," *Inf. Sci.*, vol. 231, pp. –64– 82, 2013. doi: 10.1016/j.ins.2011.08.020.

[15] Ö.A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020. doi: 10.1109/ACCESS.2019.2963724.

[16] Y. Ki, E. Kim, and H. K. Kim, "A novel approach to detect malware based on API call sequence analysis," *Int. J. Distrib. Sens. Netw.*, vol. 11, no. 6, pp. 659101, 2015. doi: 10.1155/2015/659101.

[17] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Comput. Surv. (CSUR)*, vol. 50, no. 3, pp. 1–40, 2017.

[18] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A review of android malware detection approaches based on machine learning," *IEEE Access*, vol. 8, pp. 124579–124607, 2020. doi: 10.1109/ACCESS.2020.3006143.

[19] Y. Otsuki, M. Ichino, S. Kimura, M. Hatada, and H. Yoshiura, "Evaluating payload features for malware infection detection," *J. Inf. Process.*, vol. 22, no. 2, pp. 376–387, 2014. doi: 10.2197/ipsjjip.22.376.

[20] K. Shaerpour and A. Dehghantanha, "Trends in Android malware detection," *J. Digit. Forens. Secur. Law*, vol. 8, no. 3, pp. 21, 2013. doi: 10.15394/jdfsl.2013.1149.

[21] G. S. Chhabra, V. P. Singh, and M. Singh, "Cyber forensics framework for big data analytics in IoT environment using machine learning," *Multimed. Tools Appl.*, vol. 79, no. 23–24, pp. 15881–15900, 2020. doi: 10.1007/s11042-018-6338-1.

[22] N. Idika and A. P. Mathur, "A survey of malware detection techniques," Dept. Comput. Sci., Purdue Univ., West Lafayette, 2007.

[23] L. Liu, B. S. Wang, B. Yu, and Q. X. Zhong, "Automatic malware classification and new malware detection using machine learning," *Front. Inf. Technol. Electron. Eng.*, vol. 18, no. 9, pp. 1336–1347, 2017. doi: 10.1631/FITEE.1601325.

[24] A. Mahindru and A. L. Sangal, "MLDroid—framework for Android malware detection using machine learning techniques," *Neural Comput. Appl.*, vol. 33, no. 10, pp. 5183–5240, 2021. doi: 10.1007/s00521-020-05309-4.

[25] M. Alazab, "Profiling and classifying the behavior of malicious codes," *J. Syst. Softw.*, vol. 100, no. 3, pp. 91–102, 2015. doi: 10.1016/j.jss.2014.10.031.

[26] P. Yan and Z. Yan, "A survey on dynamic mobile malware detection," *Softw. Qual. J.*, vol. 26, no. 3, pp. 891–919, 2018. doi: 10.1007/s11219-017-9368-4.

[27] J. Senanayake, H. Kalutarage, and M. O. Al-Kadri, "Android mobile malware detection using machine learning: A systematic review," *Electronics*, vol. 10, no. 13, pp. 1606, 2021. doi: 10.3390/electronics10131606.

[28] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust intelligent malware detection using deep learning," *IEEE Access*, vol. 7, pp. 46717–46738, 2019. doi: 10.1109/ACCESS.2019.2906934.

[29] A. Azmoodeh, A. Dehghantanha, and K. K. R. Choo, "Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning," *IEEE Trans. Sustain. Comput.*, vol. 4, no. 1, pp. 88–95, 2018. doi: 10.1109/TSUSC.2018.2809665.

[30] M. Rhode, P. Burnap, and K. Jones, "Early-stage malware prediction using recurrent neural networks," *Comput. Secur.*, vol. 77, no. 2, pp. 578–594, 2018. doi: 10.1016/j.cose.2018.05.010.

[31] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K. K. R. Choo, "A deep recurrent neural network based approach for internet of things malware threat hunting," *Future Gener. Comput. Syst.*, vol. 85, no. 4, pp. 88–96, 2018. doi: 10.1016/j.future.2018.03.007.

[32] D. Puthal, S. Nepal, R. Ranjan, and J. Chen, "Threats to networking cloud and edge datacenters in the internet of things," *IEEE Cloud Comput.*, vol. 3, no. 3, pp. 64–71, 2016. doi: 10.1109/MCC.2016.63.

[33] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "MalDozer: Automatic framework for android malware detection using deep learning," *Digit. Invest.*, vol. 24, no. 4, pp. S48–S59, 2018. doi: 10.1016/j.diin.2018.01.007.
[34] Q. D. Ngo, H. T. Nguyen, V. H. Le, and D. H. Nguyen, "A survey of IoT malware and detection methods based on static features," *ICT Express*, vol. 6, no. 4, pp. 280–286, 2020. doi: 10.1016/j.icte.2020.04.005.
[35] M. Tom, "Mitchell: Machine learning," *Burr Ridge*, vol. 45, no. 37, pp. 870–877, 1997.
[36] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, 2003.
[37] E. M. Dovom, A. Azmoodeh, A. Dehghantanha, D. E. Newton, R. M. Parizi, H. Karimipour, "Fuzzy pattern tree for edge malware detection and categorization in IoT," *J. Syst. Archit.*, vol. 97, no. 7, pp. 1–7, 2019. doi: 10.1016/j.sysarc.2019.01.017.
[38] F. A. Gers *et al.*, "Learning to forget: Continual prediction with LSTM," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, 1999. doi: 10.1162/089976600300015015.
[39] J. Jeon, J. H. Park, and Y. S. Jeong, "Dynamic analysis for IoT malware detection with convolution neural network model," *IEEE Access*, vol. 8, pp. 96899–96911, 2020. doi: 10.1109/ACCESS.2020.2995887.
[40] V. Rey, P. M. S. Sánchez, A. H. Celdrán, and G. Bovet, "Federated learning for malware detection in IoT devices," *Comput. Netw.*, vol. 204, no. 7, pp. 108693, 2022. doi: 10.1016/j.comnet.2021.108693.
[41] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, 2016. doi: 10.1109/TNNLS.2016.2582924.
[42] M. A. Haq and M. A. R. Khan, "DNNBoT: Deep neural network-based botnet detection and classification," *Comput. Mater. Contin.*, vol. 71, no. 1, pp. 1729–1750, 2022. doi: 10.32604/cmc.2022.020938.
[43] M. A. Haq, M. A. R. Khan, and A. H. Talal, "Development of PCCNN-based network intrusion detection system for EDGE computing," *Comput. Mater. Contin.*, vol. 71, no. 1, pp. 1769–1788, 2022. doi: 10.32604/cmc.2022.018708.
[44] M. A. Haq, "DBoTPM: A deep neural network-based botnet prediction model," *Electronics*, vol. 12, no. 5, pp. 1159, 2023. doi: 10.3390/electronics12051159.
[45] C. S. Yadav, "Malware analysis in IoT & android systems with defensive mechanism," *Electronics*, vol. 11, no. 15, pp. 2354, 2022. doi: 10.3390/electronics11152354.
[46] M. A. Haq, M. A. R. Khan, and M. Alshehri, "Insider threat detection based on NLP word embedding and machine learning," *Intell. Autom. Soft Comput.*, vol. 33, no. 1, pp. 619–635, 2022. doi: 10.32604/iasc.2022.021430.
[47] A. Kumar, "Optimal cluster head selection for energy efficient wireless sensor network using hybrid competitive swarm optimization and harmony search algorithm," *Sustain. Energy Technol. Assess.*, vol. 52, no. 4, pp. 102243, 2022. doi: 10.1016/j.seta.2022.102243.
[48] E. Balamurugan, "Network optimization using defender system in cloud computing security based intrusion detection system with game theory deep neural network (IDSGT-DNN)," *Pattern Recognit. Lett.*, vol. 156, no. 2, pp. 142–151, 2022. doi: 10.1016/j.patrec.2022.02.013.