# MTHAEL: Cross-Architecture IoT Malware Detection Based on Neural Network Advanced Ensemble Learning

Danish Vasan, Mamoun Alazab, Sitalakshmi Venkatraman, Junaid Akram, and Zheng Qin

**Abstract**—The complexity, sophistication, and impact of malware evolve with industrial revolution and technology advancements. This article discusses and proposes a robust cross-architecture IoT malware threat hunting model based on advanced ensemble learning (MTHAEL). Our unique MTHAEL model using stacked ensemble of heterogeneous feature selection algorithms and state-of-the-art neural networks to learn different levels of semantic features demonstrates enhanced IoT malware detection than existing approaches. MTHAEL is the first of its kind that effectively optimizes recurrent neural network (RNN) and convolutional neural network (CNN) with high classification accuracy and consistently low computational overheads on different IoT architectures. Cross-architecture benchmarking is performed during the training with different architectures such as ARM, Intel80386, MIPS, and MIPS+Intel80386 individually. Two different hardware architectures were employed to analyze the architecture overhead, namely Raspberry Pi 4 (ARM-based architecture) and Core-i5 (Intel-based architecture). Our proposed MTHAEL is evaluated comprehensively with a large IoT cross-architecture dataset of 21,137 samples and has achieved 99.98 percent classification accuracy for ARM architecture samples, surpassing prior related works. Overall, MTHAEL has demonstrated practical suitability for cross-architecture IoT malware detection with low computational overheads requiring only 0.32 seconds to detect Any IoT malware.

**Index Terms**—Internet-of-Things, malware threat hunting, robust malware detection, advanced ensemble learning, cross-architectures

## 1 Introduction

Internet-of-Things (IoT) refers to an extensive network of smart internet-connected devices such as Embedded systems, internet-connected home appliances, sensors and actuators that automatically sense and process the collected data. Today, IoT applications in a civilian setting could include domains such as energy management [1], [2], [3], [4], intelligent transport systems [5], [6], [7], ambient aided living [8], [9], [10], supply chain management [11], [12], healthcare [13], [14], [15] and smart grid power transmission [16]. IoT devices are empowered with limited computation, storage, and communication capabilities and more importantly, they are inherently deficient in smart device security. Therefore, despite the growing presence of IoT devices, they form an easy target to malicious attacks [17], [18]. For instance, malware such as "Hijme" [19], and "Mirai" [20] affect smart IoT devices that cause a devastating impact on smart environments of Industry 4.0.


- *Danish Vasan, Junaid Akram, and Zheng Qin are with the School of Software, Tsinghua University, Beijing 100084, China. E-mail: danish.wasan@gmail.com, znd15@mails.tsinghua.edu.cn, thuqinzheng@163.com.*
- *Mamoun Alazab is with the College of Engineering, IT & Environment, Charles Darwin University, Casuarina, NT 0810, Australia. E-mail: alazab.m@ieee.org.*
- *Sitalakshmi Venkatraman is with the Information Technology, Melbourne Polytechnic, Preston, VIC 3072, Australia. E-mail: sitavenkat@melbournepolytechnic.edu.au.*


*Manuscript received 17 Jan. 2020; revised 9 June 2020; accepted 26 July 2020. Date of publication 11 Aug. 2020; date of current version 8 Oct. 2020. (Corresponding author: Zheng Qin.) Digital Object Identifier no. 10.1109/TC.2020.3015584*

Several research studies have proposed novel data mining techniques for malware detection and this research topic continues to be active due to escalating malware attacks over Internet [21], [22], [23], [24], [25]. However, only recently, the research focus has shifted to IoT devices that run on customized operating systems. Due to the inherent limitations in resource, memory and computational power of IoT devices, existing security solutions are not well-suited over large-scale IoT networks. Majority of new IoT malware exploit low-level vulnerabilities of compromised devices to infect the IoT networks. Thus, there arises a compelling need to focus on effective solutions for IoT specific malware detection [26]. This forms the main motivation for this research work in taking the initial steps towards filling the gap in literature. The objective of this paper is to propose an efficient ensemble learning model by innovatively enhancing existing baseline architectures reported in literature and to address their shortcomings.

It is a prime requirement to identify the strengths and weaknesses of existing approaches as many learning models are shallow for IoT malware detection. With the significant increase in the number of malware attacks recently, a technique called deep learning is gaining importance and it outperforms traditional malware detection methods [27], [28], [29], [30]. By combining static and dynamic OpCode features of Android APKs, a Deep Belief Network (DBN) model reported in literature has achieved an overall malware detection accuracy of 96.76 percent [29], which could be improved with a better optimization of the deep learning model. Since malware attackers rename variables to create code confusion to fail such feature-based detection models,

new techniques such as convolutional neural networks (CNN) with visualization techniques in Android APKs are emerging [31]. However, visualization methods are still in their infancy and do not achieve superior accuracy in malware detection. Further, these studies are specific to Android APKs only, which is a major limitation. There is a need to consider other different IoT platforms in smart city infrastructure that are potentially at risk due to the potential increase in cyber-attacks.

In order to cover a wide IoT landscape of Industry 4.0, many different types of features are considered for malware analysis such as energy consumption patterns [32], OpCodes and their sequences [33], [34]. However, existing research works have predominantly studied a single feature-based machine learning technique for IoT malware detection. Some researchers have studied the efficacy of different machine learning algorithms such as Neural Networks, Decision Trees, Support Vector Machines, Bayesian Networks and K-Nearest Neighbors [35], [36]. Such studies have demonstrated malware detection accuracies ranging between 94 percent and 96 percent using OpCode features such as sequence of codes and frequency of their occurrence. However, such techniques take more processing time, and a novel, faster and light-weight learning method is warranted for the IoT landscape in order to achieve real-time security in smart cities of the future.

Over the past decade, several fast methods to train the machine learning classifiers such as AdaBoost and Support Vector Machines (SVM) have been explored [37], [38]. In particular, the Principal Component Analysis (PCA) technique to reduce the dimensionality [39] of feature vectors has become popular. Commonly used features are based on different OpCode sequences of malware and benign applications, including the recent visualization techniques for pattern recognition [40]. However, these studies were not conducted for detecting IoT malware. While OpCode analysis are more popular in achieving high accuracy, an optimal light-weight learning algorithm is necessary to be proposed and tested for a real-time detection of IoT malware efficiently. Due to the inherent computational limitations of IoT devices, we posit that a new ensemble light-weight deep learning approach using OpCode analysis would be a best-fit for IoT malware detection, which has not been explored so far. To address these shortcomings, we propose a novel IoT malware threat hunting algorithm called MTHAEL (Malware Threat Hunting based on Advanced Ensemble Learning) using stacking ensemble of heterogeneous feature selection algorithms and deep neural network architectures.

An improved predictive performance can be achieved using advanced ensemble learning by combining the results from multiple classification models into one high-quality classifier [12]. While stacking ensemble models are being explored more recently, our novel MTHAEL model aims to resolve the challenges associated with architecture overheads by using different networks for the IoT malware detection problem with limited and unevenly distributed samples. Several heterogeneous feature selection algorithms and network architectures are modelled in MTHAEL to learn the features at different semantic levels, thereby enabling the characterization of the distinct and subtle differences between malware and benign applications. Our objective is to achieve a very high accuracy as close as 100 percent in detecting malware and benign samples by incurring least computational overheads.

### 1.1 Contribution

In this paper, we propose a novel advanced ensemble-based learning approach for IoT malware threat hunting. Our model uses state-of-the-art technology in deep learning with heterogeneous feature selection algorithms to detect obfuscated malware that includes both metamorphic and polymorphic malware and achieves the highest detection so far reported in literature. To the best of our understanding, our proposed model is the first of its kind as it uniquely combines heterogeneous feature selection algorithms and network architectures for learning different levels of semantic feature representation to accurately detect cross-architecture IoT malware. Further, we demonstrate the robustness of MTHAEL algorithm against different adversarial datasets such as junk code, polymorphic malware and benign OpCode insertions in various IoT architectures with a large dataset of IoT malware samples reported so far. Another key aspect is the performance comparison of MTHAEL with existing OpCode-based malware detection systems using baseline networks reported in previous research studies [36], [37], [41]. Further, the ARM-based dataset employed for experimental analysis of MTHAEL has not been commonly adopted in literature. The aim of our in-depth and comprehensive experimental study is to show that MTHAEL detection rate can reach very close to the desired 100 percent accuracy. Furthermore, we perform tests to prove that MTHAEL can easily immune the IoT networks from obfuscation attacks with significant accuracy and least overheads. An additional advantage of our proposed robust MTHAEL model is that its novel ensemble learning approach could also be adopted for traditional non-IoT platforms.

In summary, the main contributions of this paper are:

1) A novel cross-architecture MTHAEL model is proposed to employ different feature selection techniques such as Information Gain and OpCode Dictionary to overrule less important OpCodes in order to resist various obfuscations against adversaries.
2) Advanced ensemble technique is adopted such that the sub-networks (weak-learners) are embedded in a larger multi-headed network (meta-model), for learning the best way to combine the predictions from each input sub-network efficiently.
3) A comprehensive bench-marking of MTHAEL is performed on different processor architectures using a large IoT dataset to establish an unprecedented high accuracy and robustness in detecting obfuscated malware against adversaries.
4) A unique experimental study is conducted to evaluate the architecture overhead using different hardware architectures such as Raspberry Pi 4 (ARM-based CPU), Core-i5, Core-i3 and Dual-Core for establishing MTHAEL's very low memory and processing speed overheads.

5) A rare collection of a large IoT malware dataset that covers commonly adopted cross-architecture IoT platforms is developed for the experimental setup.
6) A comparative evaluation of MTHAEL against baseline IoT networks is conducted to establish its efficiency.

The organization of the paper is as follows. In Section 2, we provide a review of related work. Section 3 presents our proposed MTHAEL model. The experimental evaluation, results, analysis and discussion are provided in Section 4. Finally, Section 5 presents the conclusion of this study and future research directions.

## 2 Related Work

Malware detection methods fall under two major categories such as static analysis and dynamic analysis [42], [43]. The commonly explored static malware detection approaches in literature are: signature-based detection, n-gram analysis and OpCode analysis that examine a software code statically to identify any malicious code. On the other hand, in dynamic malware analysis, a virtual environment (e.g., virtual machine or sandbox) is used to analyze the run-time behavior of an application program such as execution path, required operating system (OS) resources and requested privileges in order to classify if an executable is benign or malicious. Several studies have conducted surveys on both static and dynamic approaches, reporting on their advantages and disadvantages [44], [45], [46]. However, new malware variants easily evade detection by obfuscating code and behavior patterns to stay below the radar and, therefore, dynamic mobile malware detection methods have been on the rise [47] [27].

Researchers have considered novel methods such as Access Miner [48] for analyzing how OS resources are used by a program to identify normal applications and detect anomalous behavior in real-time. While such an approach does not require any training of malware samples, the accuracy of classification is only about 90 percent. An improved approach called "Deep-sign" has achieved 98.6 percent accuracy by training deep neural networks which produced invariant representations [49]. Using a sandbox to record the behavior attributes as a bit-string and inputting this into a deep 8-layered neural network, a signature of the program was generated for achieving a highly successful malware detection. However, the training time with just 1,200 input training samples had taken two days, and all the 600 testing samples considered for experimental analysis were not related to the IoT domain.

A large-scale malware analysis, classification, and clustering system called "AMAL" with rich set of static and dynamic features for generating highly-representative features had achieved a precision of 99.5 percent and recall of 99.6 percent for the classification of only certain malware families with a precision and recall of about 98 percent for unsupervised clustering [50]. While the study has benchmarked with 115,157 samples for computing the distance measures alone, the overall computational time was not studied. Another study demonstrated a hybrid approach by combining a recurrent model using multi-layer perceptron with a standard classifier such as logistic regression to improve the true positive rate to reach as high as 98.3 percent with a 0.1 percent false positive rate [51]. Hence, researchers started to explore hybrid approaches to surpass the accuracy of malware detection as compared to popular individual approaches [52]. However, these studies are for mainstream malware and not for IoT malware.

One of the early malware studies in the IoT domain used a random forest classifier and a dataset of smart internet-connected devices and resulted in an exceptionally high 99.9 percent accuracy of malware classification. A comparative experimental study with different tree-size showed that a classifier with 40 tree-size remained optimal and achieved a mean square root error of 0.0171 [53]. More recently, deep learning frameworks have been explored in a variety of cyber security applications [38], [43]. Among deep learning approaches, recurrent neural network (RNN) models with long short-term memory (LSTM) have been adopted to successfully detect malware based on OpCodes for ARM-based IoT applications [41], [54], [55]. Such studies have shown promising results using OpCode sequence analysis reaching high levels of efficiency when deep learning-based approaches were adopted for malware detection. However, there is scope for improvement in existing malware detection models for IoT.

IoT devices are different to personal computers and laptops as they are equipped with diverse CPU architectures and possess unique characteristics such as limited resources, continuous online connection requirement and lack of security protection. Hence, existing malware detection models being resource exhaustive are not suitable for IoT malware detection. In addition, a variety of IoT devices are being manufactured resulting in diverse processor architectures such as MIPS, ARM, PowerPC, and SPARC. Hence, Virtual sandbox based on QEMU can be used to monitor executable files. QEMU supports about 26 different CPU architectures, including IoT processor architectures such as MIPS and ARM and supports different operating systems such as Windows and Linux. IoTPOT and Detux are the main virtual sandboxes employed in this study to perform dynamic analysis of malware.

Overall, we find from literature that OpCode-based features are highly efficient for malware detection, and not much research studies have explored IoT malware despite witnessing an increasing trend in IoT exploits and threats recently. In addition, there is a scarcity of research in adopting ensemble learning for robust IoT malware threat hunting. In this paper, we seek to contribute towards filling this gap in literature. We explore the potential use of OpCode-based features with an ensemble of different network architectures based on deep learning frameworks for achieving an efficient detection of IoT malware. We combine different network architectures to learn different levels of semantic feature representation and show that an ensemble of neural networks can detect the cross-architecture IoT malware more accurately.

## 3 Proposed MTHAEL Approach

### 3.1 Overview

We propose a novel cross-architecture approach called MTHAEL (Malware Threat Hunting Based on Advanced Ensemble Learning) which employs an advanced ensemble technique with heterogeneous feature selection algorithms using feature representations of OpCode Instructions to

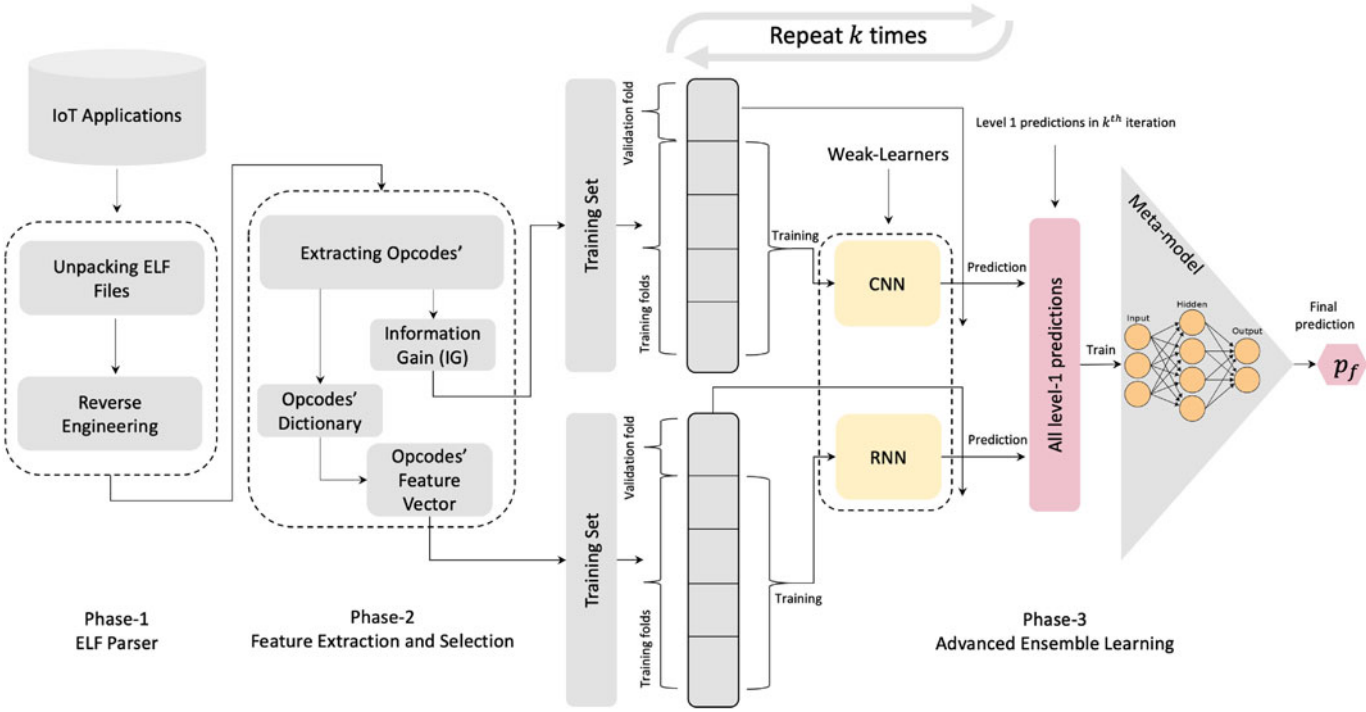


Fig. 1. High-level architectural view of MTHAEL based on advanced ensemble technique.

determine whether an application in an IoT platform has a malware threat or not. Fig. 1 shows a high-level architectural view of the proposed MTHAEL model based on advanced ensemble technique. The main processor architectures used today are: 32-bit (x86) and 64-bit (x86-64, IA64, and AMD64). These processor architectures vary in the data path width, integer size, and memory address width. The diversity in processor architectures is even more prominent in the IoT landscape with varied resource limitations. Hence, we design our MTHAEL approach with the aim that it caters to this diversity in IoT architectures. The novelty of MTHAEL is to learn from several weak-learners and combine them by training a strong-learner to output enhanced predictions based on the multiple predictions returned by these weak models.

The detailed processes involved in our proposed MTHAEL model are grouped under three main phases as given below:

1) Executable and Linkable Format (ELF) Parsing,
2) Feature Extraction and Selection, and
3) Advanced Ensemble Learning.

In Phase 1, a disassembler such as Object-Dump tool is utilized to extract the OpCodes to disassemble the dataset of IoT applications. The features of OpCode Instructions are obtained by extracting OpCode sequence from each executable file in assembly format as illustrated in Fig. 2. Typically, an executable file in assembly format contains many predefined segments, such as .init segment, .finit segment and .text segment. The .text segment stores program instructions, while the rest of segments are for storing data. Hence, the contents of .text segment alone are used to extract all OpCode sequences as they represent the logic of the corresponding ELF file. The pseudocode of our OpCode extraction algorithm is given in Algorithm 1.

Phase 2 of MTHAEL involves feature extraction and selection. It is important to understand IoT applications consisting of long sequences of OpCodes which are instructions

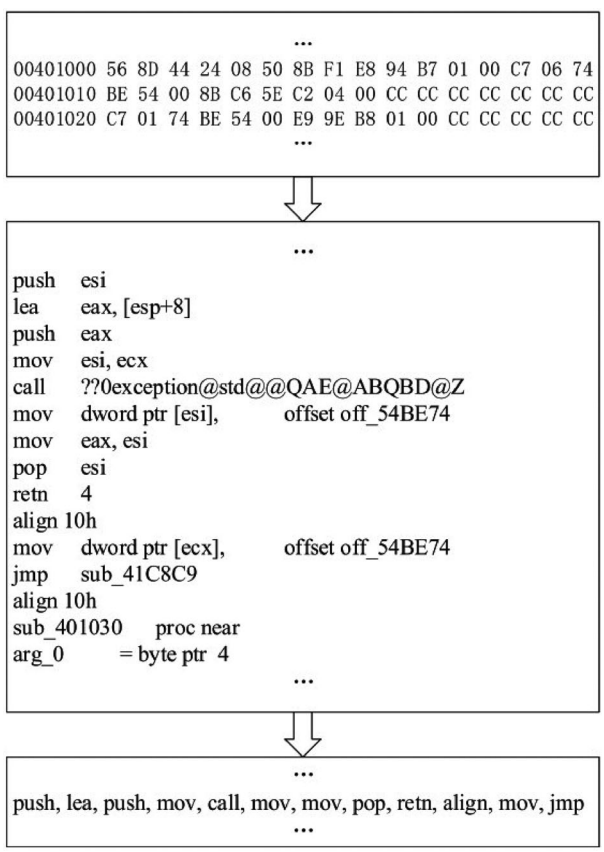


Fig. 2. An example of IoT OpCode sequence extraction.

to be performed on an IoT device processing unit. Creating an n-gram OpCode sequence as features is a common approach to classify malware based on their disassembled codes [56]. The number of rudimentary features for code length $L$ is $S^L$, where $S$ is the size of instruction set. It is clear that a huge increase in $L$ will result in feature explosion. In addition, by decreasing the size of features optimally, the robustness and effectiveness of detection can be increased since ineffective features that reduce the performance of machine learning approach are removed. Therefore, in order to avoid feature explosion, MTHAEL employs two different types of feature selection techniques (Information Gain in CNN and OpCode Dictionary in RNN) to reduce the feature set by determining the best features [57].

Information retrieval techniques are widely used for the feature selection process [58]. In the context of resource constrained IoT platforms warranting an efficient feature selection technique, we propose to adopt an information-theoretic approach called Information Gain (IG) defined in Equation (1). Using IG, features are ranked based on the amount of information content available for classification in order to select global features. With IG, more useful features are selected to overcome imperfections in global feature selection.

**Algorithm 1.** Extract OpCode Sequence

**Input:** Each assembly format file
**Output:** Corresponding OpCode sequence

```
Pattern ←predefine matching pattern for extracting OpCode
file ← open (assembly format file)
for eachline in file do
  if eachline start with '.text' then
    result ← match(pattern,eachline)
    if result is not null then
      add result to corresponding OpCode sequence
    end if
  end if
end for
```

Next, we introduce slightly different feature selection technique called OpCode dictionary which contains OpCode instructions extracted from the OpCode sequences. Our OpCode dictionary based approach eliminates the repetitive OpCodes and builds a heterogeneous OpCode feature vector. We make use of this OpCode feature vector to train the sub-network(RNN) at level-1. The main benefit of this approach is, it reduces the size of the feature vector which is beneficial for faster network training. This process is depicted in Fig. 3.

The mathematical definition of IG is given in Equation (1), where $f$ is the feature in a dataset $D$, $k$ is the number of classes in the training set, $D_v$ is the OpCode stream where feature $f$ exists, and $w_i$ is the proportion of $D_v$ to class $i$.

$$IG(D,f) = \sum_{i=1}^{k} -p_i \ \ln\ p_i - \sum_{v\in\{0,1\}} \frac{D_v}{D} \sum_{i=1}^{k} -w_i \ \ln\ w_i \tag{1}$$

The values of IG are calculated for each feature and sorted in decreasing order to identify the most prominent features required in the setting of a threshold ($\alpha > 0.2$).

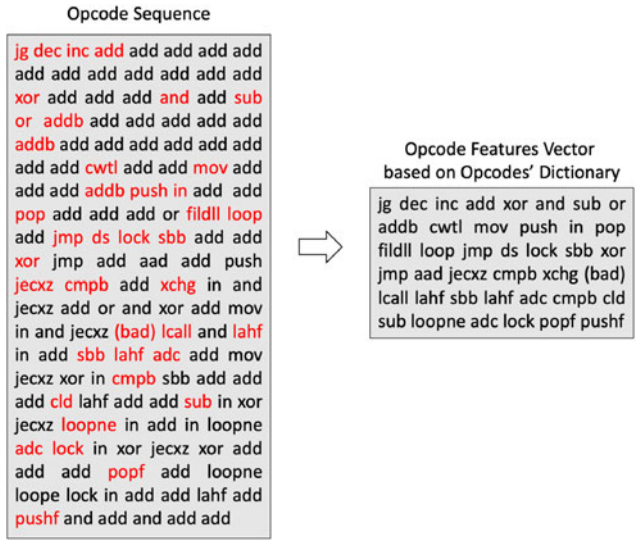


Fig. 3. An extraction of non-repetitive OpCode Instructions from the OpCode sequences.

Hence, for Phase 2 of MTHAEL we have employ both IG and OpCode Dictionary for feature selection as the best-fit approach in the case of IoT applications.

In Phase 3, the advanced ensemble technique embeds the sub-networks into a large multi-headed network, which learns how to best combine the predictions from each input sub-network. During the classification, two feature selection techniques are applied on test dataset that includes both IoT malware and benign files. We adopt two different neural network architectures, namely Recurrent Neural Network (RNN), and Convolutional Neural Network (CNN) in order to make the best use of the potential unique capabilities of each different architecture. Next, we describe RNN and CNN employed in Phase 3 of our MTHAEL model.

### 3.2 Recurrent Neural Network (RNN)

A recurrent neural network is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence [59]. This allows it to exhibit a temporal dynamic behavior. Unlike feed-forward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. The basic building block in RNN is a Recurrent Unit (RU). There are many different variants of RU such as Long-Short-Term-Memory (LSTM) and Gated Recurrent Unit (GRU) which are used in our algorithm. Simple variants may perform better than both LSTM and GRU [60]. However, they are not implemented in Keras which is used in this research uniquely to enable fast experimentation with deep neural networks.

Fig. 4 shows the abstract idea of RU, which has an internal state that is being updated every time the unit receives a new input. A gate in RU is responsible for regulating the information by mapping old state values to the new state value. In addition, another gate is used for calculating the output values of RU. The implementation of these gates varies for different types of RU. The LSTM has more gates than the GRU but some of the redundant gates are omitted in our proposed model. In order to train the RU, the weight-

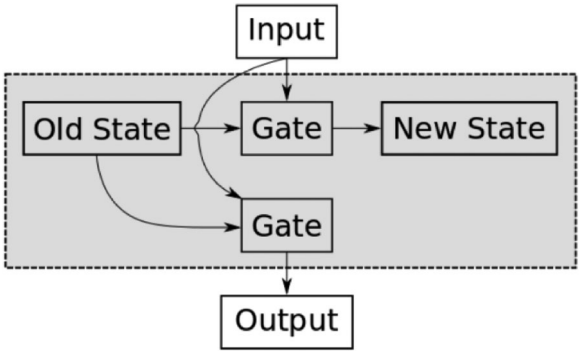


Fig. 4. An abstract idea of RU (Recurrent Unit).

matrices of the gates are gradually changed so that the RU gives the desired output for an input sequence. We adopt TensorFlow to automate this process.

### 3.3 Convolutional Neural Network (CNN)

Many researchers focus on two-dimensional CNN for image recognition problems while studies using 1DCNNs can be found in the literature in the context of natural language processing (NLP) [61] . A CNN works well for identifying simple patterns within the data which can then be used to form more complex patterns within higher layers. On the other hand, 1DCNN could be used to derive interesting features from shorter (fixed-length) segments of the overall dataset and where the location of the feature within the segment is not of high relevance. CNNs share the same characteristics and follow the same approach, whichever type they belong to, namely 1D, 2D or 3D. The critical difference is the dimensionality of the input data and how the feature detector (or filter) slides across the data. Hence, in the interest of limited resources in IoT devices and considering the overheads involved, we adopt 1DCNN in MTHAEL. The repetitive operations and the associated mathematical formulas used in MTHAEL are presented below.

- *Convolution Operations:* The convolution operations reduce the number of features while maintaining the key features such as interpretation invariance, rotation invariance, and scale invariance. This layer decreases the over-fitting significantly and also improves the generalization ability of the architecture. In this layer, the input consists of several maps, and the output results in maps after size reduction. The resulting output map represents a combination of convolution values of input maps [62], and is expressed by the following equation:

$$x_j^l = f\left(\sum_{i \in M_j} x_j^{l-1} * k_{ij}^l + b_j^l\right), \tag{2}$$

where $M_j$ represents the group of input maps, $k_{ij}^l$ denotes the convolution kernel used for joining the $i^{\text{th}}$ input feature map with $j^{\text{th}}$ output feature map, $b_j^l$ represents the bias corresponding to $i^{\text{th}}$ feature map and $f$ is the activation function used. We measured the sensitivity by the equation given below:

$$\delta_j^l = \delta_j^{l+1}\ w_j^{l+1}\ of\ (u^l) = \beta_j^{l+1}\ up\left(\delta_j^{l+1}\right) of(u^l), \tag{3}$$

where the $l+1$ layer is the pooling layer, the weight $w$ represents a convolution kernel, and its value $\beta_j^{l+1} up(.)$ representing an up-sampling operation. The partial derivative of the error cost function used with bias $b$ and convolution kernel $k$ are defined as follows:

$$\frac{\partial E}{\partial b_j} = \sum_{s,t}\left(\delta_j^l\right) s,t \tag{4}$$

$$\frac{\partial E}{k_{ij}^l} = \sum_{s,t}\left(\delta_j^l\right) s,t\ (p_i^{l-1})\ s,t, \tag{5}$$

where $(p_j^{l-1})s,t$ is the patch for each convolution of $x_j^{l-1}, k_{ij}^l,\ (s,t)$, as the center of the patch, and its location value is obtained by convolution of the patch of the $s,t$ location in the input feature and the convolution kernel $k_{ij}^l$.

- *Pooling Operations:* The pooling operations are generally known to constitute the subsampling layer. Typically, it contains two types of pooling, namely maximum pooling and average pooling, and are not affected by backward propagation. This layer minimizes the effect of image deformation, i.e., translation and scale rotation. It also aids in reducing the dimension of the feature map and maximizing the model performance. For outputting of each sampling, we define the feature map used is defined as follows:

$$x_j^l = f\left(down(x_j^{l-1}) + b_j^l\right), \tag{6}$$

where $down(.)$ denotes a pooling task and $b$ represents the bias. We computed the sensitivity as follows:

$$\delta_j^l = \delta_j^{l+1}\ w_j^{l+1}\ of\ (u). \tag{7}$$

The partial derivative of the error cost function with respect to bias $b$ is given as follows:

$$\frac{\partial E}{\partial b_j} = \sum_{u,v}\left(\delta_j^l\right) u,v. \tag{8}$$

Also, the weight formula was upgraded for the classification task.

- *Fully connected layer:* The output generated from the previous operations such as convolution layers and pooling layers are then classified by a fully connected layer. Each neuron of an earlier layer is then connected with the neuron of a fully connected layer.

### 3.4 Activation Function (Softmax)

The activation function used in the implementation of our MTHAEL model is the Softmax classifier. Softmax is a generalized logistic function emphasizing the essential values of vectors while blocking those with values lower than maximum. Several CNN architectures employ Softmax function

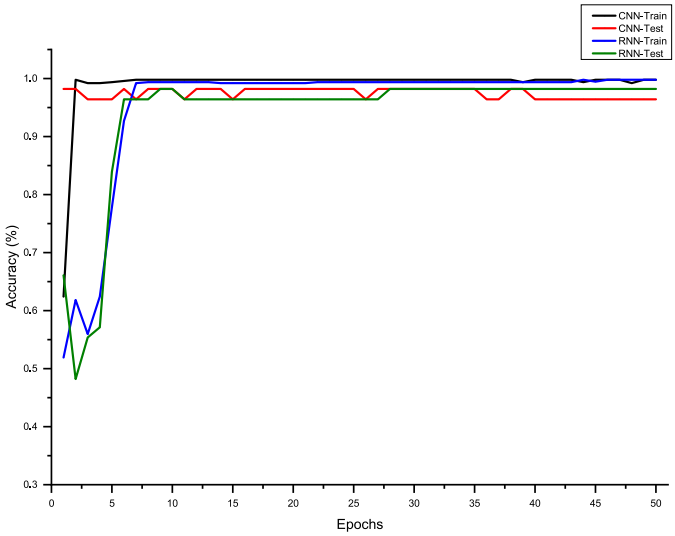


Fig. 5. An average training and testing accuracy for weak-learners.

in the classification layer [61], [63], [64], [65]. Since Softmax is the core element used in deep learning classification tasks, we have employed it to output a normalized form of its inputs.

Figs. 5 and 6 show the trends in two measures obtained with weak-learners over 50 epochs: i) average training and testing accuracy, and ii) average training and testing loss. One can see that after 40 epochs, the average training and testing accuracy as well as average training and testing loss with weak-learners were stabilized for both RNN and CNN. Similarities of testing and training curves suggest that our trained weak-learners did not overfit training data. The total training statistics of all the weak-learners and meta-learner are summarized in Table 1 which shows the number of parameters required for training over 50 epochs.

### 3.5 MTHAEL Design

We designed our MTHAEL model for cross-architecture IoT malware detection, with a hybrid of RNN and CNN forming the sub-networks. We extracted all the OpCodes from the IoT-based malware and benign applications of the training dataset and trained RNN and CNN using a Softmax classifier. For efficient training, we overruled less important

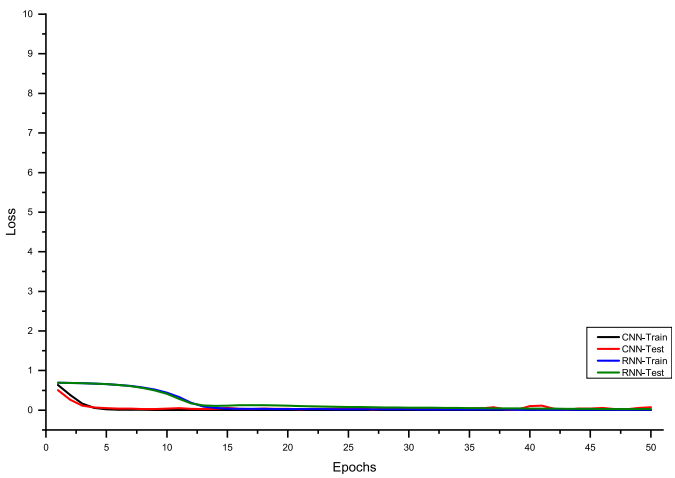


Fig. 6. An average training and testing loss for weak-learners.

TABLE 1
Training Statistics

| Architecture | Total Parameters | Trainable Parameters | Non-Trainable Parameters |
|---|---|---|---|
| RNN | 10,744 | 10,744 | 0 |
| CNN | 756,856 | 756,856 | 0 |
| MTHAEL | 767,667 | 37 | 767,630 |

OpCodes. Further, in order to resist junk-code insertion attacks, we applied efficient heterogeneous feature selection approaches such as OpCode dictionary and IG [66]. We embedded these sub-networks in a larger multi-headed neural network for learning how to best combine the predictions from each input sub-network. It allowed the stacking ensemble to be treated as a single large model. The benefit of this approach was the outputs of the sub-networks were provided directly to the meta-model with an additional capability to update the weights of the sub-networks for achieving optimized results. More formally, Algorithm 2 summarizes the stacking ensemble learning with cross-validation technique that was employed in MTHAEL model.

**Algorithm 2.** MTHAEL With k-Fold Cross Validation

**Input:** Training Data $D = \{x_i, y_i\}_{i=1}^{m} (x_i \in \Re_n, y_i \in \gamma)$
**Output:** An Advanced Ensemble Classifier $H$

Step 1: Adopt cross validation approach in preparing a training set for seconds level classifier
Randomly split $D$ in $K$ equal-sizes subsets: $D = D1, D2, D3, ...., D_k$
**for** $k \leftarrow 1$ to $K$ **do**
  Step 1.1: Learn first-level classifiers
  **for** $t \leftarrow 1$ to $T$ **do**
    Learn a classifier $h_{k_1}$ from $D/D_k$
  **end for**
  Step 1.2: Construct a training set for second-level classifier
  **for** $x_i \in D_k$ **do**
    Get a record $\{x_i^{'}, y_i\}$, where $x_i^{'} = \{h_{k_1}(x_i), h_{k_2}(x_i), \ldots, h_{k_r}(x_i)\}$
**end for**
**end for**
Step 2: Learn a second-level Classifier
Learn a new classifier $h^{'}$ from the collection of $\left\{x_i^{'}, y_i\right\}$
Re-learn first-level classifiers
**for** $t \leftarrow 1$ to $T$ **do**
  Learn a classifier $h_t$ based on $D$
**end for**
**return** $H(x) = h(h_1(x), h_2(x), \ldots, h_r(x))$

## 4 Experimental Evaluation

### 4.1 IoT Dataset

IoT datasets are being developed recently and public collections of IoT malware samples have not yet matured in terms of diversity and size. Due to these constraints, related studies reported in literature are limited by the small IoT dataset with additional drawbacks of being unbalanced. Further, a comprehensive experimental investigation with a large collection of cross-architecture dataset for both training and testing has not been the norm in the evaluation of existing

works. Hence, possible arguments exist that previously proposed IoT malware detection models may work only with their specific IoT datasets. One of the aims of this work is to address this limitation in literature.

We collected cross-architecture IoT malware from many sources such as IoTPoT [67], Detux [68] and Virus-Share [69]. Next, we filtered only executable ELF files and verified with Virus-Total [70]. Similarly, benign samples were collected from more than 23,000 firmware router images [71] such as Asus, Belkin, Tenvis, Dlink, TP Link, Linksys, Trendnet, Centurylink, Zyxel and Openwrt. These benign samples were from the predominantly used Intel 80386 and X86-64 platforms with new installations of Ubuntu OS along with standard PC-based applications. We ascertained the balance in cross-platform representations such that both malware and benign samples were spread across common IoT architectures such as MIPS, ARM, PowerPC and PC-based architectures like Intel X86-64. Table 2 provides the distribution of sample dataset across various architectures that were considered for our experimental study. The 'Shared' column in Table 2 gives the number of samples common to all three sources (Virus-Share, IoTPoT, Detux), which is below 5 percent of the total malware samples. This shows that the malware dataset collected from the three sources are almost independent of each other with least overlap. While many researchers have not given importance in having a large dataset of benign samples, a balanced dataset is required to benchmark the robustness and effectiveness of malware detection. By eliminating these limitations of previous studies, we provide a large IoT dataset that is balanced along with high cross-architecture representation.

TABLE 2
IoT Dataset Statistics Information

| Architecture | Virus-Share | IoTPot | Detux | Shared | Total Malware | Total Benign |
|---|---|---|---|---|---|---|
| ARM | 2,506 | 912 | 35 | 26 | 3,427 | 2035 |
| Intel80386 | 7,100 | 570 | 29 | 5 | 7,694 | 1438 |
| X86-64 | 1,099 | 320 | 11 | 3 | 1,427 | 283 |
| MIPS | 18,15 | 935 | 282 | 98 | 2,934 | 1899 |
| | 12,520 | 2,737 | 357 | 132 | 15,482 | 5,655 |

### 4.2 Experimental Setup and Environment

We adhered to the standard norms and guidelines of an experimental setup that is required in the domain of malware detection. We adopted k-fold cross validation technique with k=10 to evaluate our predictive models and considered the average of the results for comparing the performance of MTHAEL. For benchmarking, we compared the performance of MTHAEL with various existing architectures for malware detection that have used the same OpCode sequences and dataset. The experimental program was written in Python, and the hardware environment used Intel Core-i5 (9th Generation) processor with 8-GB RAM for training. Further, we used different processors with different configurations of low resource IoT devices to evaluate the architecture overhead of MTHAEL in testing its practical suitability for IoT platforms.

### 4.3 Performance Measurements

Standard measures such as true positive (TP), true negative (TN), false positive (FP) and false negative (FN) were calculated. The well-known metrics such as Accuracy, Precision, Recall or TPR (true positive rate), and F1-score were measured. These metrics have been accepted by the research community to provide a rigorous evaluation of any proposed method in literature [72], [73], [74]. We summarize the formula for these key metrics used in our experimental evaluation of MTHAEL as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}. \tag{12}$$

### 4.4 Performing Single Comparison With Other Baseline Networks on ARM-Based IoT Dataset

In our first set of experiments, we focused only on the ARM-based IoT dataset collection with 5,462 ARM samples of 3,427 malware and 2,035 benign samples for testing the performance of our proposed MTHAEL model. A standards-based experimental benchmarking was conducted to evaluate the performance of the proposed advanced ensemble approach used in MTHAEL as compared to the two baseline networks, RNN, and CNN using the same feature selection techniques. We trained all baseline networks for 50 epochs with the same hyperparameter settings. The average classification accuracies achieved for each model with the test dataset are: 98.21 percent for RNN, 96.43 percent for CNN, and 99.98 percent for MTHAEL.

A comprehensive performance evaluation of MTHAEL for ARM-based IoT malware detection was conducted using standards metrics as compared to the RNN, and CNN architectures for all the baseline networks. Table 3 compares the classification accuracy and other standard measures of MTHAEL algorithm to the baseline networks (RNN and CNN). The results prove that MTHAEL approach is superior than all the baseline networks with higher accuracies achieved consistently throughout the experimental study. MTHAEL has proven to demonstrate 1.77 percent higher accuracy than the RNN baseline network and 3.55 percent higher accuracy than the CNN baseline network. Table 3 provides a comparison of all the models with respect to the standard metrics such as Accuracy, Precision, Recall-rate, and F1-score.

Overall, the time taken to classify a new IoT malware for each type of IoT device was determined by going through the three phases of our proposed MTHAEL model: i) extracting the OpCodes from dissembled binary, ii) selecting the most important OpCode instruction from extracted OpCode based on feature selection techniques, and iii) predicting whether malware or benign by the trained model. Our experimental

TABLE 3
Performance Comparison of Cross-Architecture MTHAEL Model versus Baseline Networks on ARM-Based IoT Dataset

| Metrices | Architecture | | | | | |
|---|---|---|---|---|---|---|
| | RNN | | CNN | | MTHAEL | |
| | Benign | Malware | Benign | Malware | Benign | Malware |
| **Accuracy (%)** | 98.21 | | 96.43 | | 99.98 | |
| **F1-Score (%)** | 98.46 | 97.87 | 96.96 | 95.65 | 99.95 | 99.93 |
| **Precision (%)** | 96.96 | 100 | 94.11 | 100 | 99.96 | 99.95 |
| **Recall (%)** | 100 | 95.83 | 100 | 91.66 | 100 | 99.95 |

TABLE 4
Overhead Comparison of Cross-Architecture MTHAEL Model on Different Processor Architectures Using ARM-Based IoT Dataset

| Metrices | Machine | | | |
|---|---|---|---|---|
| | Raspberry Pi 4 with 4GB RAM | Core-i5 with 8GB RAM | Core-i3 with 4GB RAM | Dual-Core with 4GB RAM |
| **Memory (MiB)** | $442^{+}_{-}$ | $447^{+}_{-}$ | $447^{+}_{-}$ | $449^{+}_{-}$ |
| **Average Prediction Time (Seconds)** | 0.32 | 0.10 | 0.21 | 0.43 |

TABLE 5
Evaluation of Cross-Architecture MTHAEL Model on Different IoT Architecture Datasets

| IoT Dataset | | Accuracy (%) | FPR (%) | FNR (%) |
|---|---|---|---|---|
| Trained by | Evaluated by | | | |
| ARM | X86-64 | 97.27 | 1.01 | 2.49 |
| Intel80386 | MIPS | 95.72 | 2.81 | 2.56 |
| MIPS | Intel80386 | 58.2 | 1.12 | 82.3 |
| MIPS+ Intel80386 | MIPS+ Intel80386 | 97.02 | 1.05 | 2.51 |

TABLE 6
A Summary of Classification Accuracy (%) of Proposed MTHAEL Compared to Baseline Networks Against Junk-Code Insertion Attacks

| Architecture | Percentage | | | | | |
|---|---|---|---|---|---|---|
| | 5% | 10% | 15% | 20% | 25% | 30% |
| RNN | 98.10 | 97.53 | 97.01 | 96.35 | 96.65 | 96.11 |
| CNN | 96.05 | 95.21 | 94.32 | 93.87 | 93.84 | 93.25 |
| MTHAEL | 99.86 | 99.05 | 98.71 | 98.15 | 98.12 | 97.98 |

TABLE 7
Polymorphic Dataset

| $\lambda$ ratio | Dataset Name |
|---|---|
| 0% | $D_{morphed_{0\%}}$ |
| 25% | $D_{morphed_{25\%}}$ |
| 50% | $D_{morphed_{50\%}}$ |
| 75% | $D_{morphed_{75\%}}$ |
| 100% | $D_{morphed_{100\%}}$ |

TABLE 8
A Summary of Classification Accuracy (%) of Proposed MTHAEL Model Compared to Baseline Networks Against Polymorphic Malware Attacks

| Dataset Name | MTHAEL | RNN | CNN |
|---|---|---|---|
| $D_{morphed_{0\%}}$ | 100 | 99.45 | 99.25 |
| $D_{morphed_{25\%}}$ | 100 | 99.05 | 98.92 |
| $D_{morphed_{50\%}}$ | 98.19 | 97.11 | 96.35 |
| $D_{morphed_{75\%}}$ | 96.11 | 95.10 | 95.51 |
| $D_{morphed_{100\%}}$ | 94.53 | 92.13 | 92.21 |

results show that MTHAEL model takes on an average only 0.32 seconds with Raspberry Pi 4, 0.10 seconds with Core-i5 (9th Generation) processor, 0.21 seconds with Core-i3 (4th Generation) processor, and 0.43 seconds with Dual-Core processor. These promising results suggest that MTHAEL is efficient and suitable for real-time large IoT deployments such as smart cities of the future. Table 4, presents a summary of the comparative study conducted on memory and prediction time overheads with the proposed MTHAEL model using two different hardware architectures such as Raspberry Pi 4 (ARM-based processors) and Intel processors.

Based on the insights gained from the experimental results, it can be seen that MTHAEL model exhibits the highest F1-score of 99.94 percent, compared to RNN with 98.16 percent, and CNN with 96.30 percent. In addition, MTHAEL model remained superior against different adversarial attacks demonstrating high accuracy achieved as compared to baseline network models. Tables 6 and 8 present the summary of different adversarial attacks of code obfuscations by injecting junk codes, polymorphic malware, and benign OpCodes. The details of testing the robustness of MTHAEL are provided in the subsection on Robustness Evaluation with Adversaries.

### 4.5 Evaluation of Cross-Architecture Samples

In the next set of experiments, we considered cross-architecture samples with the aim of degrading the effectiveness of the MTHAEL-based scores. First, we trained our proposed cross-architecture MTHAEL model with ARM-based IoT dataset. Next, in this experiment, we tested with X86-64 dataset which was collected within our complete IoT dataset.

Table 5 show the experimental results of our proposed crossed-architecture MTHAEL model. When we trained our model by the ARM dataset and evaluated with the X86-64 dataset, it achieved a high accuracy of 97.27 percent and an affordable FPR of 1.01 percent. This result is attributed to the fact that X86-64 dataset contains the same type of malware except some malware instances of X86-64 that do not appear in the ARM dataset.

For our next experiment, we trained our proposed cross-architecture MTHAEL model on Intel80386 dataset and tested with the MIPS dataset. For this experiment, we achieved the acceptable accuracy of 91.82 percent with affordable FPR of 5.7 percent. On the other hand, when we trained MTHAEL on MIPS dataset and tested with Intel80386 dataset, the results were not favourable. We observe that the Intel 80386 dataset contains more types of malware than the MIPS dataset, and only some malware instances of Intel 80386 appear in MIPS dataset also. Thus, the size of dataset representing a wide

range of malware types plays a major role in training the model to achieve efficiency in detection. Experimental results also prove that we could detect zero-day malware on a new IoT architecture by learning with malware samples from existing and common architectures like Intel 80386.

### 4.6 Detection on Mixed Architecture Samples

In the next set of experiments, we evaluated the performance of MTHAEL using a dataset with samples collected from mixed architectures. We prepared a mixed architecture dataset from Intel80386 and MIPS samples for training and testing our MTHAEL model. The experimental results are presented in Table 5 with an accuracy of 97.02 percent, FPR of 1.05 percent and FNR of 2.51 percent. Not many existing studies have performed mixed architecture to possibly compare with the performance of MTHAEL comprehensively. However, we observe that MTHAEL has achieved a higher accuracy as compared to an accuracy of 85.2 percent reported in a recent work despite tested on a larger mixed architecture dataset [75].

### 4.7 Robustness Evaluation With Adversaries

Malware authors adopt circumvention techniques such as metamorphic and polymorphic obfuscations to prevent being detected by anti-malware solutions. Previous studies [38] show the use of obfuscation methods that pose major threats falling under two broad categories of obfuscations as summarised below:

1) *Metamorphic obfuscation* changes the code itself with no need to use encryption while retaining the same functionality. The four commonly used techniques for metamorphic obfuscation are described below:
   - *Dead-code Insertion* or trash insertion does nothing to code logic but changes the byte string of the code making it difficult to be detected. Dead-code insertion using complicated operation code sequences are applied if the code or function has no effect and such an approach is adopted to change the virus signature to some extent. Examples of such dead-codes are [NOPs: No Operation Performed], [MOV eax, eax], [SHL eax, 0], [ADD eax, 0] and [INC eax] followed by [DEC eax], and these not only pass values using registers but also through memory.
   - *Code Transposition*, shuffles instruction sequences completely changing the order of binary content. Examples include JMP and CALL instructions in which the order of instructions are different from original ones. Code transpositions such as subroutine outlining, subroutine inlining and subroutine permutation are used to change the order of instructions.
   - *Register Reassignment* replaces codes between registers through register name exchange without affecting other behaviors of the program. For instance, if register ebx is assumed to be dead throughout a certain live range of register eax, it can replace eax within that live range. Another example is replacing [PUSH ebx] with [PUSH eax] to exchange register names.
   - *Instruction Substitution* uses a dictionary of equivalent sequences of instructions for replacing instruction sequences. Some examples are given below:

     [MOV EAX, ECX] equivalent to [XOR EBX, EAX], or [XOR EAX, EAX] equivalent to [MOV EAX, 0]
2) *Polymorphic obfuscation* applies encryption and data appending/data pre-pending for changing malware body and changes decryption routines after each infection if encryption keys are changed. This leads to creation of signatures that prevent infections very difficult to be detected. Packing is very popular for code compression or obfuscation. A packer is a software capable of being used for compressing and encrypting PE in secondary memories and restoring original executable images when loaded into main memories (RAM). Polymorphic packers can make their "signatures" mutate. This method is the most popular obfuscation techniques employed by malware writers and distributors.

Statistic reports and literature suggest that more than 80 percent of malware binaries use obfuscation methods to avoid being detected. It was also found that OpCode-based malware detection could be resilient for obfuscation methods [38], [41], [76]. Neural Network based approach is gaining importance to detect this kind of adversarial attacks.

In this experimental study, we evaluated the robustness of MTHAEL model by employing adversarial attacks with the aim of degrading the effectiveness of MTHAEL-based scores to determine the limitations of the model.

- First, we considered a set of experiments by introducing junk code instructions into the ARM IoT dataset. A similar approach was employed by previous researchers [41].
- Second, we considered a set of experiments by introducing a polymorphic malware into the ARM IoT dataset. A similar approach was employed by previous researchers [76].
- Third, we considered a set of experiments by introducing benign OpCodes into the ARM IoT dataset. To the best of our knowledge, this kind of approach has never been studied before.

In the first set of adversary attacks to test the robustness of MTHAEL model against junk code insertion attacks, we introduced OpCode sequences using instructions such as NOP that do not actually make any effect on the program semantics. This is aimed at degrading the effectiveness of the MTHAEL-based scores to test its limitations and boundaries of performance. Similar techniques have previously been adopted to defeat many statistics-based and machine learning based scores [41], [77]. Junk code insertion technique would obfuscate malicious OpCode sequences and reduce the balance of malicious OpCodes in a malware.

Here, we use an affinity based criteria to evade junk OpCode injection anti-forensics technique. Our heterogeneous feature selection algorithms have the advantage of automatically eliminating less instructive OpCodes to mitigate the effects of injecting junk OpCodes as compared to other methods.

To demonstrate robustness of MTHAEL against code insertion attack, in an iterative manner, a specified proportion (5, 10, 15, 20, 25, 30 percent) of all elements in each sample's generated graph were selected randomly and their value incremented by one. For example, in the 4th iteration of the evaluations, 20 percent of the indices in each sample's graph were chosen to increment their value by one. In addition, in our evaluations the possibility of a repetitive element selection was included to simulate injecting an OpCode more than once. Incrementing $E_j^i$ in the sample's generated graph is equivalent to injecting $OpCode_j$ next to the $OpCode_i$ in a sample's instruction sequence to mislead detection algorithm. Algorithm 3 describes an iteration of junk code insertion during experiments and it is necessary to mention this procedure should repeat for each iteration of k-fold validation.

**Algorithm 3.** Pseudocode for Junk-Code Insertion

**Input:** Trained Classifier $D$, Test Samples $S$, Junk Code Percentage $k$
**Output:** Predicted Class for Test Samples $P$
$P = \{\}$
**for** each sample in $S$ **do**
  $W =$ Compute the Opcodes of $sample$
  $R =$ { select $k\%$ of $W^{'}$s (index randomly)}
  **for** each $index$ in $R$ **do**
    $W_{index} = W_{index} + 1$
  **end for**
  Normalized $W$
  $P = P \bigcup D(W)$
**end for**
**return** $P$

Table 6 presents the performance of the MTHAEL model under the junk code insertion attacks. From the experimental results, it can be seen that MTHAEL has the highest accuracy then the baseline networks. These results have established that MTHAEL can achieve an acceptable performance for detecting junk code injection attacks and has shown to be superior to other methods.

**Algorithm 4.** Pseudocode for Polymorphic Malware

**Input:** Dissembled sample is given
**for** each line in dissembled **do**
  //Rule #1
  **if** line match to 'Mov R#, 0' **then**
    replace it to 'EOR R#$_1$,R#$_2$'
  **end if**
  //Rule #2
  **if** line match to 'MUL R#$_1$, R#$_2$ R#$_3$' **then**
    replace it to 'MLA R#$_1$, R#$_2$ R#$_3$'
  **end if**
  //Rule #3
  **if** line match to 'MVN R#$_1$,R#$_2$' **then**
    replace it to 'mov R$_n$,o ORN R$_1$,R$_n$,R$_2$'
  **end if**
  //Other rules
**end for**

In the second set of adversary attacks to test the robustness of MTHAEL, we test the model with polymorphic version of IoT malware dataset. For building the polymorphic malware samples, we used an open-source polymorphic malware creation tool similar to Algorithm 4 that is publicly available in portals such as GitHub.com for obfuscating ARM-based binaries. We created five datasets based on ARM-based IoT dataset as shown in Table 2 that included 100 polymorphic samples in each dataset. The polymorphic malware creation tool replaces an OpCode or a sequence of OpCodes with functionally equivalent OpCodes, and $\lambda$ specifies the percentage of instructions that satisfies polymorphism rules. Table 7 shows the different polymorphic datasets considered with $\lambda$ values of 0, 25, 50, 75 and 100 percent. For this experiment, we evaluated our ARM-based trained MTHAEL model with these polymorphic datasets that resulted in a superior overall classification accuracy than other baseline sub-networks. The detailed results of this second set of adversarial experiments are summarized in Table 8.

TABLE 9
A Summary of Classification Accuracy (%) of Proposed Cross-Architecture MTHAEL Model and Baseline Networks Against Benign-Code Insertion Attacks

| Architecture | Original | Adding Benign OpCodes |
|---|---|---|
| RNN | 98.21 | 95.77 |
| CNN | 96.43 | 92.89 |
| MTHAEL | 99.98 | 98.11 |

**Algorithm 5.** Pseudocode for Benign-Code Insertion

**Input:** Trained Classifier $D$, Test Samples $S$, Benign Sample $B$
**Output:** Predicted Class for Test Samples $P$
$P = \{\}$
**for** each sample in $S$ **do**
  $W =$ Compute the Opcodes of $sample$
  $N_{append} = W + B$
**end for**
Normalized $N$
$P = P \bigcup D(N)$
**return** $P$

In the third set of adversary attacks, we studied the robustness of MTHAEL by introducing benign OpCode insertion attacks. As per our understanding this kind of approach has never been studied in previous literature. In order to achieve this, the entire code from one extraneous benign sample was introduced into all test samples similar to Algorithm 5. Through this experimental study we find that an overall classification accuracy of 95.77 percent was achieved for RNN, and 92.89 percent for CNN. Adversarial attacks using such benign OpCodes have significantly declined the detection accuracy of RNN and CNN as compared to our proposed MTHAEL, which achieved only a slight decline with an acceptable high accuracy of 98.11 percent. A summary of the experimental results of benign OpCode injection is presented in Table 9.

Overall, the above experimental results indicate that MTHAEL-based scoring is reasonably robust against various adversarial attacks using obfuscations that are typically found in malwares. More importantly, MTHAEL has demonstrated resistance against recent obfuscated malwares that have an increasing trend to target multi-architectures of IoT.

TABLE 10
Performance Comparison of the Proposed Cross-Architecture MTHAEL Model versus Existing OpCode Based Classification Techniques

| Algorithm | Dataset | Year | Accuracy(%) | Precision(%) | Recall(%) | F1-Score(%) |
|---|---|---|---|---|---|---|
| Santosh *et al.* [36] | Windows | 2013 | 95.91 | 86.25 | 77.70 | 81.75 |
| Hashemi *et al.* [37] | Windows | 2017 | 96.87 | 91.09 | 81.55 | 86.05 |
| N-gram [78] | Windows | 2018 | 93.21 | - | 89.22 | - |
| LSTM+CNN+MF(MalNet) [78] | Windows | 2018 | 99.88 | - | 99.14 | - |
| Azmoodeh *et al.* [32] | ARM-based IoT | 2018 | 91.19 | 83.33 | 94.20 | 88.44 |
| Azmoodeh *et al.* [41] | ARM-based IoT | 2018 | 99.68 | 98.59 | 98.37 | 98.48 |
| HaddadPajouh *et al.* [55] | ARM-based IoT | 2018 | 94 | - | - | - |
| **MTHAEL** | **ARM-based IoT** | **-** | **99.98** | **99.96** | **99.97** | **99.94** |

### 4.8 Comparison With Existing Opcode-Based Malware Detection Technique

Finally, we compared the performance of the proposed cross-architecture MTHAEL model with existing malware classification studies that have used OpCode based feature selection techniques for IoT-based malware classification using conventional machine learning and deep learning methods. We considered popular classification techniques from recent existing research that have adopted the same three-phase process: i) extract OpCode sequences from the IoT applications, ii) apply the features selection techniques to obtain the important features, and iii) input the extracted features to the machine learning or deep learning classifiers for further classification. Table 10 presents an overall performance comparison of MTHAEL model versus other existing OpCode based classification models. In order to demonstrate fairness in comparison of our proposed MTHAEL approach with previously proposed approaches reported in literature, we considered prior models that were developed based on similar OpCodes sequences as features. We considered four studies that used Windows based dataset and three studies that used the same ARM-based IoT dataset. These studies were performed with limited number of samples as compared to the sample size in our experimental study with MTHAEL. From Table 10, we find that our proposed cross-architecture (MTHAEL) model is superior to popularly adopted existing malware classification models in all the standard metrics used for benchmarking.

## 5 Conclusion and Future Work

With an unprecedented growth in IoT, it is predicted that the constant race between the cyber attackers and cyber defenders could play a major impact on the advancement of Industry 4.0 in realising smart cities of the future. Hence, it calls for a robust IoT malware threat hunting solution for providing secure smart city infrastructures. This paper proposed a new Malware Threat Hunting based on Advanced Ensemble Learning (MTHAEL) for cross-architecture IoT malware detection with the aim of achieving maximum accuracy while maintaining least overheads. By uniquely embedding a hybrid of RNN and CNN sub-networks in a larger multi-headed neural network, MTHAEL was designed to perform deep learning by optimally combining the predictions from each input sub-network. This resulted in developing a more accurate anti-malware detection solution than the individual sub-networks. Another unique contribution is our unique creation of a diverse large IoT dataset consisting of multi-architecture samples including adversarial OpCode obfuscations of binaries. This cross-architecture IoT dataset addressed the gap in the dataset imbalance, which is one of the limitations in the experimental evaluation methods used in related works found in literature. Using such a large IoT dataset, we conducted systematic and comprehensive set of experiments to evaluate and benchmark our proposed MTHAEL model for the classification and detection of IoT malware on each architecture as well as mixed architectures. The performance metrics of MTHAEL were benchmarked with existing baseline models that used similar OpCode feature based detection approaches with the same samples taken from our data subsets for performing a fair comparison. In addition, the robustness of MTHAEL was evaluated on cross-architectures to detect obfuscated malware using adversarial attacks.

Overall, the experimental evaluation of the proposed MTHAEL approach using our large IoT dataset achieved good results with 99.98 percent detection accuracy for ARM-based malware. For experiments with mixed-architecture, MTHAEL demonstrated a high detection accuracy of 97.02 percent, while its cross-architecture training and testing led to varying detection performances that provided insights into the malware trends, limitations and new findings. We conclude that MTHAEL is effective in detecting zero-day malware on a new IoT architecture by training on malware samples from existing common architectures like Intel 80386. In all these experiments, MTHAEL consistently outperformed existing baseline approaches in detection accuracy as well as computational overheads.

We envisage extensions of this study as part of our future research work in the following directions: (1) to conduct experiments evaluating MTHAEL for large-scale IoT applications and multi-class prediction with a specific focus in improving the classification techniques for a more diverse set of adversarial attacks of IoT; (2) to explore and determine the relationships of malware instances and attack trends in different IoT architectures; and (3) to investigate novel deep learning approaches in the stackable ensemble learning of MTHAEL for achieving further improvements in the model.

## Acknowledgments

This work was supported by the Department of Corporate and Information Services, NTG.

## REFERENCES


[1] Z. U. Shamszaman, S. Lee, and I. Chong, "WoO based user centric energy management system in the internet of things," in *Proc. Int. Conf. Inf. Netw.*, 2014, pp. 475–480.

[2] M. Wang, G. Zhang, C. Zhang, J. Zhang, and C. Li, "An IoT-based appliance control system for smart homes," in *Proc. Int. Conf. Intell. Control Inf. Process.*, 2013, pp. 744–747.

[3] A. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, pp. 2787–2805, 2010.

[4] F. D. Miorandi, D. Sicari, S. Pellegrini, and I. Chlamtac, "Ad Hoc networks Internet of Things: Vision, applications and research challenges," *Ad Hoc Netw.*, vol. 10, pp. 1497–1516, 2012.

[5] M. Gerla, E. K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *IEEE World Forum Internet Things*, 2014, doi: 10.1109/WF-IoT.2014.6803166.

[6] K. Su, J. Li, and H. Fu, "Smart city and the applications," in *Proc. Int. Conf. Electron. Commun. Control*, 2011, pp. 1028–1031.

[7] Z. Xiong, H. Sheng, W. G. Rong, and D. E. Cooper, "Intelligent transportation systems for smart cities: A progress review," *Sci. China Inf. Sci.*, vol. 55, pp. 2908–2914, 2012.

[8] A. J. Jara, M. A. Zamora, and A. F. G. Skarmeta, "An architecture based on Internet of Things to support mobility and security in medical environments," in *Proc. 7th IEEE Consum. Commun. Netw. Conf.*, 2010, pp. 1–5.

[9] A. J. Jara, M. A. Zamora, and A. F. G. Skarmeta, "An Internet of Things-based personal device for diabetes therapy management in ambient assisted living (AAL)," *Pers. Ubiquitous Comput.*, vol. 15, pp. 431–440, 2011.

[10] X. M. Zhang and N. Zhang, "An open, secure and flexible platform based on Internet of Things and cloud computing for ambient aiding living and telemedicine," in *Proc. Int. Conf. Comput. Manage.*, 2011, pp. 1–4.

[11] A. Dada and F. Thiesse, "Sensor applications in the supply hain: The example of quality-based issuing of perishables," in *Proc. Internet Things*, 2008, pp. 140–154.

[12] Y. Gu and T. Jing, "The IOT research in supply chain management of fresh agricultural products," in *Proc. 2nd Int. Conf. Artif. Intell.*, 2011, pp. 7382–7385.

[13] M. Bazzani, D. Conzon, A. Scalera, M. A. Spirito, and C. I. Trainito, "Enabling the IoT paradigm in E-health solutions through the VIRTUS middleware," in *Proc. 11th IEEE Int. Conf. Trust*, 2012, pp. 1954–1959.

[14] C. Doukas, I. Maglogiannis, V. Koufi, F. Malamateniou, and G. Vassilacopoulos, "Enabling data protection through PKI encryption in IoT m-Health devices," in *Proc. IEEE 12th Int. Conf. Bioinf. Bioengineering*, 2012, pp. 25–29.

[15] R. S. H. Istepanian, A. Sungoor, A. Faisal, and N. Philip, "Internet of M-health Things m-IoT," in *Proc. IET Seminar Assisted Living*, 2011, pp. 1–3.

[16] M. Yun and B. Yuxin, "Research on the architecture and key technology of Internet of Things (IoT) applied on smart grid," in *Proc. Int. Conf. Advances Energy Eng.*, 2010, pp. 69–72.

[17] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Internet of Things security and forensics: Challenges and opportunities," *Future Gener. Comput. Syst.*, vol. 78, pp. 544–546, 2018.

[18] A. Mosenia and N. K. Jha, "A comprehensive study of security of Internet-of-Things," *IEEE Trans. Emerg. Top. Comput.*, vol. 5, no. 4, pp. 586–602, Oct.–Dec. 2017.

[19] G. Kambourakis, C. Kolias, and A. Stavrou, "The Mirai botnet and the IoT zombie armies," in *Proc. IEEE Mil. Commun. Conf.*, pp. 267–272, 2017.

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

[21] J. Gardiner and S. Nagaraja, "On the security of machine learning in malware C8C detection," *ACM Comput. Surv.*, vol. 49, 2016, Art. 59.

[22] J. Peng, K. K. R. Choo, and H. Ashman, "User profiling in intrusion detection: A review," *J. Netw. Comput. Appl.*, vol. 72, pp. 14–27, 2016.

[23] E. M. Rudd, A. Rozsa, M. Günther, and T. E. Boult, "A survey of stealth malware attacks, mitigation measures, and steps toward autonomous open world solutions," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 1145–1172, Second Quarter 2017.

[24] S. Iqbal *et al.*, "On cloud security attacks: A taxonomy and intrusion detection and prevention as a service," *J. Netw. Comput. Appl.*, vol. 74, pp. 98–120, 2016.

[25] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Comput. Surv.*, vol. 50, 2017, Art. no. 41.

[26] Z. K. Zhang, M. C. Y. Cho, C. W. Wang, C. W. Hsu, C. K. Chen, and S. Shieh, "IoT security: Ongoing challenges and research opportunities," in *Proc. IEEE 7th Int. Conf. Service-Oriented Comput. Appl.*, 2014, pp. 230–234.

[27] P. Yan and Z. Yan, "A survey on dynamic mobile malware detection," *Softw. Qual. J.*, vol. 26, pp. 891–991, 2018.

[28] Z. M. Fadlullah *et al.*, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2432–2455, Fourth Quarter, 2017.

[29] 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.

[30] S. Venkatraman, M. Alazab, and R. Vinayakumar, "A hybrid deep learning image-based analysis for effective malware detection," *J. Inf. Secur. Appl.*, vol. 47, pp. 377–389, 2019.

[31] Y. S. Yen and H. M. Sun, "An Android mutation malware detection based on deep learning using visualization of importance from codes," *Microelectron. Reliab.*, vol. 93, pp. 109–114, 2019.

[32] 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, pp. 1141–1152, 2018.

[33] E. M. Karanja, S. Masupe, and J. Mandu, "Internet of Things malware. A survey," *Int. J. Comput. Sci. Eng. Surv.*, vol. 8, no. 3, pp. 1–20, Jul. 2017.

[34] D. Bilar, "Opcodes as predictor for malware," *Int. J. Electron. Secur. Digit. Forensics*, vol. 1, pp. 156–168, 2008.

[35] R. Moskovitch *et al.*, "Unknown malcode detection using OPCODE representation," in *Proc. Eur. Conf. Intell. Secur. Inform.*, 2008, pp. 204–215.

[36] 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.: Int. J.*, vol. 231, pp. 64–82, 2013.

[37] H. Hashemi, A. Azmoodeh, A. Hamzeh, and S. Hashemi, "Graph embedding as a new approach for unknown malware detection," *J. Comput. Virol. Hacking Tech.*, vol. 13, pp. 153–166, 2017.

[38] M. Alazab, S. Venkatraman, P. Watters, M. Alazab, and A. Alazab, "Cybercrime: The case of obfuscated malware," in *Proc. Int. Conf. e-Democracy*, 2012, pp. 204–211.

[39] M. Siddiqui, M. C. Wang, and J. Lee, "Data mining methods for malware detection using instruction sequences," in *Proc. IASTED Int. Conf. Artif. Intell. Appl.*, 2008, pp. 358–363.

[40] S. Venkatraman and M. Alazab, "Use of data visualisation for zero-day malware detection," *Secur. Commun. Netw.*, vol. 18, 2018, Art. no. 1728303.

[41] 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, Mar. 2019.

[42] K. Shaerpour, A. Dehghantanha, and R. Mahmod, "Trends in android malware detection," *J. Digit. Forensics Secur. Law*, 2017, doi: 10.15394/jdfsl.2013.1149.

[43] G. S. Chhabra, V. P. Singh, and M. Singh, "Cyber forensics framework for big data analytics in IoT environment using machine learning," *Multimedia Tools Appl.*, vol. 79, pp. 15881–15900, 2020.

[44] N. Idika and A. P. Mathur, "A survey of malware detection techniques," *SERC Tech. Reports*, 2007.

[45] S. Peng *et al.*, "Survey on malware detection methods," in *Proc. 3rd Hackers' Work. Comput. Internet Secur.*, 2009, doi: 10.1145/1327452.1327492.

[46] Z. Bazrafshan, H. Hashemi, S. M. H. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," in *Proc. 5th Conf. Inf. Knowl. Technol.*, 2013, pp. 113–120.

[47] M. Alazab, "Profiling and classifying the behavior of malicious codes," *J. Syst. Softw.*, vol. 100, pp. 91–102, 2015.

[48] A. Fattori, A. Lanzi, D. Balzarotti, and E. Kirda, "Hypervisor-based malware protection with AccessMiner," *Comput. Secur.*, vol. 52, pp. 33–50, 2015.

[49] O. E. David and N. S. Netanyahu, "DeepSign: Deep learning for automatic malware signature generation and classification," in *Proc. Int. Joint Conf. Neural Netw.*, 2015, pp 1–8.

[50] A. Mohaisen, O. Alrawi, and M. Mohaisen, "AMAL: High-fidelity, behavior-based automated malware analysis and classification," *Comput. Secur.*, vol. 52, pp. 251–266, 2015.

[51] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2015, pp. 1916–1920.
[52] S. Huda, J. Abawajy, M. Alazab, M. Abdollalihian, R. Islam, and J. Yearwood, "Hybrids of support vector machine wrapper and filter based framework for malware detection," *Futur. Gener. Comput. Syst.*, vol. 55, pp. 376–390, 2016.
[53] M. S. Alam and S. T. Vuong, "Random forest classification for detecting android malware," in *Proc. IEEE Int. Conf. Green Comput. Commun., IEEE Internet Things IEEE Cyber Physical Social Comput.*, 2013, pp. 663–669.
[54] M. Rhode, P. Burnap, and K. Jones, "Early-stage malware prediction using recurrent neural networks," *Comput. Secur.*, vol. 77, pp. 578–594, 2018.
[55] 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," *Futur. Gener. Comput. Syst.*, vol. 85, pp. 88–96, 2018.
[56] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev, and Y. Elovici, "Detecting unknown malicious code by applying classification techniques on OpCode patterns," *Secur. Inform.*, vol. 1, 2012, Art. no. 1.
[57] A. Feizollah, N. B. Anuar, R. Salleh, and A. W. A. Wahab, "A review on feature selection in mobile malware detection," *Digital Investigation: Int. J. Digital Forensics Incident Response*, vol. 13, pp. 22–37, 2015.
[58] G. Forman, "An extensive empirical study of feature selection metrics for text classification george," *J. Mach. Learn. Res.*, vol. 3, pp. 1289–1305, 2003.
[59] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn.*, 2015, pp. 2342–2350.
[60] Y. D. Prabowo, H. L. H. S. Warnars, W. Budiharto, A. I. Kistijantoro, Y. Heryadi, and Lukas, "LSTM and simple RNN comparison in the problem of sequence to sequence on conversation data using Bahasa Indonesia," in *Proc. 1st Indonesian Assoc. Pattern Recognit. Int. Conf.*, 2019, pp. 51–56.
[61] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2012.
[62] J. Bouvrie, "Notes on convolutional neural networks," in *Pract.*, 2006, [Online]. Available: http://dx.doi.org/10.1016/j.protcy.2014.09.007
[63] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.
[64] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[65] C. M. Bishop, "Pattern recognition and machine learning (information science and statistics)," *Mach. Learn.*, 1st ed., 2006. 2nd printing ed., New York, NY, USA: Springer-Verlag, 2006.
[66] M. R. Smith *et al.*, "Dynamic analysis of executables to detect and characterize malware," 2017, *arXiv: 1711.03947v2*.
[67] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: A novel honeypot for revealing current IoT threats," *J. Inf. Process.*, vol. 24, pp. 522–533, 2016.
[68] "GitHub — detuxsandbox/detux: The multiplatform linux sandbox," Accessed: May 07, 2020, [Online]. Available: https://github.com/detuxsandbox/detux
[69] "VirusShare.com," Accessed: May 7, 2020, [Online]. Available: https://virusshare.com/
[70] "VirusTotal," Accessed: Oct. 5, 2019, [Online]. Available: https://www.virustotal.com/gui/home/upload
[71] N. P. Tran, N. B. Nguyen, Q. D. Ngo, and V. H. Le, "Towards malware detection in routers with C500-toolkit," in *Proc. 5th Int, Conf. Inf. Commun. Technol.*, 2017, pp. 1–5.
[72] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features," in *Proc. 10th Int. Conf. Malicious Unwanted Softw.*, 2015, pp. 11–20.
[73] S. Ni, Q. Qian, and R. Zhang, "Malware identification using visualization images and deep learning," *Comput. Secur.*, vol. 77, pp. 871–885, Aug. 2018.
[74] A. P. Namanya, I. U. Awan, J. P. Disso, and M. Younas, "Similarity hash based scoring of portable executable files for efficient malware detection in IoT," *Future Gener. Comput. Syst.*, vol. 13, pp. 22–37, 2019.
[75] M. Alhanahnah, Q. Lin, Q. Yan, N. Zhang, and Z. Chen, "Efficient signature generation for classifying," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2018, pp. 1–9.
[76] H. Darabian, A. Dehghantanha, S. Hashemi, S. Homayoun, and K. K. R. Choo,"An opcode-based technique for polymorphic Internet of Things malware detection," *Concurr. Comput.*, vol. 32, no. 6, pp. 1–14, 2020.
[77] A. Walenstein and A. Lakhotia, "The software similarity problem in malware analysis," *Duplication Redundancy Similarity Softw.*, R. Koschke, E. Merlo and A. Walensteinpp, Eds., pp. 1862–4405, 2007. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2007/964
[78] J. Yan, Y. Qi, and Q. Rao, "Detecting malware with an ensemble method based on deep neural network," *Secur. Commun. Netw.*, vol. 2018, pp. 1–16, 2018.

**Danish Vasan** received the master of philosophy in computer science (MPhil-CS) from the Isra University of Pakistan, in 2014. He is working toward the PhD degree with the School of Software, Tsinghua University, Beijing China. His research interests include malware analysis, cyberSecurity, artificial intelligence, image processing, and Internet-of-Things.

**Mamoun Alazab** received the PhD degree in computer science from the Federation University of Australia, School of Science, Information Technology and Engineering. He is an associate professor with the College of Engineering, IT and Environment, Charles Darwin University, Australia. He is a cyber security researcher and practitioner with industry and academic experience.

**Sitalakshmi Venkatraman** received the MSc degree in mathematics and MTech degree in computer science from the Indian Institute of Technology, Madras, in 1985 and 1987, respectively, the PhD degree in computer science from the National Institute of Industrial Engineering, in 1993, and the MEd degree from the University of Sheffield, in 2001. She has more than 30 years of international work experience both in industry and academics, and is currently analytics discipline leader at Melbourne Polytechnic, Australia.

**Junaid Akram** received the 1st master's degree in major of information technology from UOG Pakistan, in 2009, the 2nd master's degree in major of communication and information system from ZJUT China, in 2015, and the PhD degree from the School of Software Tsinghua University China, in 2020. Since 2015, he is working as a research scholar with the Key Laboratory of Information System Security, School of Software, Tsinghua University, Beijing, China.

**Qin Zheng** is a doctoral supervisor, professor, the director of Software Engineering and Management Research Institute and Information Institute, Tsinghua University. He is the evaluation expert of National Science and Technology Award, Ministry of Education Technology Award, Major State Basic Research Development Program (973), National High Technology Research and Development Program of China (863), National Defense 10th 5-Year Plan and Ministry of Education College Undergraduate Teaching.