Computers & Security 120 (2022) 102779

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Deep learning based cross architecture internet of things malware detection and classification

Rajasekhar Chaganti[a], Vinayakumar Ravi[b,*], Tuan D. Pham[b]

[a] Dept. of Computer Science, University of Texas at San Antonio, San Antonio, Texas 78249, USA
[b] Center for Artificial Intelligence, Prince Mohammad Bin Fahd University, Khobar, Saudi Arabia

ARTICLE INFO

Article history:
Received 28 August 2021
Revised 27 April 2022
Accepted 26 May 2022
Available online 30 May 2022

Keywords:
IoT Malware
Deep learning
Recurrent neural network
Gated recurrent unit
Convolution neural network
Byte sequences
Internet of things
Malware classification

ABSTRACT

The number of publicly exposed Internet of Things (IoT) devices has been increasing, as more number of these devices connected to the internet with default settings. The devices accessed with default credentials are getting compromised with brute force attempts or the vulnerable devices are compromised with exploits to install the malware and perform malicious activity like initiating denial of service (DoS) attacks. The malware detection and classification in IoT paradigm is still considered a problem, as the adversary consistently create new variants of IoT malware and actively look for compromising the victim devices. In this paper, we proposed a Deep Learning (DL) based Bidirectional-Gated Recurrent Unit-Convolutional Neural Network (Bi-GRU-CNN) model to detect the IoT malware and classify the IoT malware families using Executable and Linkable Format (ELF, formerly named Extensible Linking Format) binary file byte sequences as an input feature. In addition, Recurrent Neural Network (RNN) based DL model combinations are considered to evaluate the performances of the IoT malware detection and family classification and also those models performance is compared with the proposed method. Our performance evaluation shows that our proposed approach obtained 100% accuracy for IoT malware detection case and 98% for IoT malware family classification. Further evaluation of our proposed model with only byte sequence as an input feature exhibit similar performance as the byte sequence and CPU types as an input features and showed that our model is robust and platform independent to detect and classify the IoT malware.

1. Introduction

The number of embedded devices connected to the internet are projected to increase to 43 billion by the end of 2023 and this number is threefold increase since 2018 (Fredrik Dahlqvist, 2019). The rapid adoption of the IoT technology for various applications such as smart home, smart industries, smart health enabled not only handheld devices but also physical devices connected to the internet. These IoT devices constantly communicate with vendor cloud to exchange the device status and other device application information to receive the constant device updates. Although IoT devices help to improve the consumer quality of life, these devices are vulnerable to security attacks and can easily be exploited if no security countermeasures taken in place. For instance, researchers discovered that 178 million IoT devices visible in public internet to the attacker in ten US cities, which include webcams, medical devices, routers, and more (Charlie Osborne, 2017). An adversary may compromise these exposed devices and use for illegitimate purpose. So, it is highly desired to secure the IoT devices and protect them from cyberattacks.

Most of the IoT devices operate on Linux based operating systems. Consequently, the malware authors try to develop the source code keeping in mind the weaknesses identified in the IoT device operating systems and applications. In addition, sometimes, the vendors don't build the IoT applications keeping security in mind and rather focus on the rapid development progress to ship the product sooner for competitive advantage. So, the adversary is just one step away to find the vulnerability and compromise the devices. Furthermore, some consumers don't give attention to securing their devices and leave the devices with default password settings. As adversary may take advantage of these default settings to compromise the devices. The mirai botnet is an example of IoT malware (Dima Ben, 2016). The variants of the malware has emerged in the wild, as the source code is released on public in 2016 (Jgamblin, 2017). Most of the IoT malware variants brute force default user credentials to access IoT devices or leverage the

* Corresponding author.
E-mail addresses: Raj.chaganti2@gmail.com (R. Chaganti), vravi@pmu.edu.sa (V. Ravi), tpham@pmu.edu.sa (T.D. Pham).

https://doi.org/10.1016/j.cose.2022.102779
0167-4048/

known vulnerabilities to exploit the devices. Once the device is compromised, an adversary may further install malware to perform activities such as acting as a bot to participate in large scale Distributed Denial of service (DDoS) attacks (Boppana et al., 2020). The commonly used conventional techniques such as signature, heuristics, and anomaly based detection mechanisms (Bazrafshan et al., 2013; Scott, 2017) can easily evade if an adversary is well versed about those solutions. The signature based solutions can only detect the known attacks (Carrillo-Mondéjar et al., 2020). The heuristics and anomaly based solutions may trigger more false positives and additional efforts required to tune the alerts. Additionally, the low power, low processing capacity and low memory IoT devices may not accommodate to include the antimalware solution at the endpoint level. Furthermore, the IoT devices operate on different operating systems and heterogeneous CPU architectures like X86, Advanced RISC Machine (ARM), PowerPC (PPC), and Microprocessor without Interlocked Pipeline Stages (MIPS). So, it is challenging to detect and classify the IoT malware using the conventional malware detection and classification techniques.

Recently, researchers proposed Machine Learning (ML) and DL based solutions for malware detection (Rathore et al., 2018; Vinayakumar and Soman, 2018). The existing work mostly focused on detecting the malware in portable executable (PE) executable files running on Windows operating systems (OS) (Vinayakumar et al., 2019a; 2019b), as the Windows OS is widely used in larger community and also windows is more prone to attacks due to attacker's motive to widespread malicious activity by infecting many devices. But, Linux based ELF binary malware binary detection received a major attention recently, as the Linux operated many IoT devices connected to the internet. ML techniques are applied to detect the ELF based malware (Shahzad and Farooq, 2012). However, ML techniques require additional efforts required for feature extraction and domain expert knowledge to obtain the essential features for malware detection. Lately, DL models are considered for solving problems in diverse fields including the malware detection (Shalaginov and Øverlier, 2021). Researchers even showed that the DL models capable of handling whole binary as an input to learn the structural and semantics of the input data (Raff et al., 2018). However, a combination of feature selection with minimal feature extraction/engineering efforts and then taking advantage of DL models may give fruitful results without having deep malware domain knowledge.

Typically, the malware binary features may be extracted using the static analysis, dynamic analysis or the image based representation of the malware. The extracted malware features may be applied to ML/DL models to detect and classify the IoT malware. But, the careful selection of the features is needed to detect and classify the malware, as the diversity of central processing unit (CPU) architectures supported by IoT malware may not capture essential information from ELF binary to categorize the malware and benignware. The static features used in the literature for ELF based IoT malware detection and classification includes Operational Code (Opcode), printable strings, Control Flow Graphs (CFG), ELF header statistics, byte sequences (Raju et al., 2021). Most of these static feature extraction process require using forensic tools to obtain the features. The dynamic features memory, process information, API call traces or network traces may also be considered for the IoT malware detection and classification. But, setting up the virtual environment for IoT malware dynamic analysis is complex as the IoT devices work on various operating systems and should support multiple CPU architectures. The image based malware representation still requires additional tools to transform the malware into an image. So, it is essential to choose the convenient, optimal feature types for IoT malware detection and classification.

The static feature opcode and malware image representation requires the ELF binary sample conversion to the other form Alrawi et al. (2021). These features may not be an optimal for malware detection in some scenarios where detection response time is critical. The CFG extraction process requires disassembling the ELF file and may require additional efforts to generate CFG, which may not be desired in some cases where the dataset is large. To overcome these limitations, we have adopted the byte sequence as a feature type in our approach to extract the few bytes from the entry point of the program in a ELF binary with minimal feature processing efforts. For packed malware, the starting point may be moved but we may not miss the classification of these samples. Consequently, the features further gone through natural language processing (NLP) preprocessing steps prior to apply various DL based models. Furthermore, we are the first to propose a bidirectional-gated recurrent unit-convolutional neural network (Bi-GRU-CNN) DL model for effective, robust, and platform independent IoT malware detection and classification on a dataset containing 8 IoT malware family feature samples and malware supported by various CPU architectures such as MIPS, ARM, X86, PowerPC. To the end, the main contributions of our work are as follows:

- Proposed Bi-GRU-CNN based DL approach for robust, CPU platform independent for cross architectural IoT malware detection and family classification.
- Utilized the byte sequence from an entry point of the ELF binary program as a feature to effectively detect and classify IoT malware with minimal feature preprocessing efforts.
- Presented the detailed performance analysis of the various DL models to compare and contrast their effectiveness for accurate IoT malware detection and classification.
- Detailed comparison of our work with the relevant prior art work in terms of performance, DL models, features, number of malware families is presented and provided insights on the gaps in literature for IoT malware detection and classification.
- t-distributed stochastic neighbor embedding (t-SNE) visualization approach was employed to ensure that the learned features were meaningful for IoT malware detection and IoT malware family classification using the proposed deep learning model.

2. Background and related work

As the number of embedded devices connecting to the internet increasing day by day, the internet users owning the IoT devices are more prone to get targeted with IoT attacks. Adversaries adopting new attack techniques and tactics to perform malicious activity. The low power, memory, and processing capable IoT devices are not an exception in the attacker list of the targeting devices. Most of the IoT devices has limited security features such as capability for fixing the vulnerabilities instantly and continuous monitoring of the malicious anomaly activity. A unified security solution is far away from reality as the diversified IoT device applications operate on different operating systems and these IoT devices work on heterogeneous CPU architectures.

IoT devices and their applications are operated with variety of operating systems such as Contiki, RIOT, TinyOS, LiteOS, FreeRTOS, Mantis OS, Nano-RK, SOS, uClinux, OpenTag, Erika Enterprise, Mbed OS, MicroPython, Embedded Linux, Windows10 IoT, and OpenWrt (Gaur and Tahiliani, 2015; PELAEZ, 2021). These number of operating systems used in IoT space makes it more difficult to track and patch the vulnerabilities on time when the IoT zero-day vulnerabilities disclosed in public. So, an adversary may leverage these vulnerabilities and quickly exploit to install the malware before fixing them. Detection of the malware exploiting the IoT devices is not easy, as the IoT devices operate on various architectures such as ARM, X86, MIPS, PowerPC, and X86-64 and performing dynamic analysis for IoT ELF files is challenging as compared to the

PE file analysis. One of the well known malware targeted the Linux based operating system is "BASHLITE", which takes the advantage of the "shellshock" vulnerability disclosed in 2014 to compromise the Linux based systems. The compromised devices include the IoT device as well and later in 2016 it was identified that 96% of these devices are IoT devices (cameras and DVR's) (Loeb, 2016). These compromised systems could generate high volume of the network traffic like 400 Gbps for performing DDoS attacks. The IoT malware received major attention in security community in 2016 when the Mirai malware source code disclosed in public (Krebs on Security, 2016). Mirai botnet is known to be one of the most notable IoT malware and has capabilities to execute various DoS attacks. Mirai uses the brute forcing attack to crack the default credentials through opened TELNET ports in public routers. A successful compromise of the IoT devices will become part of the botnet controlled by the command and control server. These IoT botnets can send attack traffic to the targeted victim, when the bot master send a trigger command. In recent years, various IoT malware families like dofloo, xorddos, tsunami targeting the IoT devices are identified by security researchers. Most of these malware were seen to be used for building botnet of IoT devices to initiate DDoS attacks.

IoT malware detection and classification using the existing malware tools has been a challenge, since most of the tools are mainly built protecting workstations, personal computers and server systems supported by X86 CPU architecture. In addition, there are number of IoT malware variants seen in the wild trying to evade the defender detection capabilities. The signature, behavior and heuristic based detection mechanisms may not be able to detect unknown attacks (Vinayakumar et al., 2019b). Recently, the ML and DL models were extensively used to detect IoT malware and classify the cross architectural IoT malware families (Dib et al., 2021; Kumar and Lim, 2019; Nghi Phu et al., 2021). In order to perform the efficient and accurate malware detection using ML models, proper feature selection plays a major role. As seen in PE malware detection using static, dynamic, and image based features (Vinayakumar et al., 2019a; 2019b), the essential IoT features can be extracted using similar methods for applying ML and DL techniques. However, the dynamic analysis feature extraction require dedicated virtual environment setup because the malware can be targeted to any one of the heterogeneous IoT supported CPU architectures. The malware features considered for applying ML and DL models in the IoT malware detection and classification literature are described in the following paragraphs by categorizing them into the static, dynamic, and hybrid methods used for feature extraction.

Static analysis features: ELF Header: The Unix based binary files are usually represented in ELF format to store the executable program bytes. The ELF header size is 52 or 64 bytes for 32 bit and 64 bit binary files (Linuxfoundation, 2021). The ELF header fields were used as a feature set to distinguish the malware and benignware in IoT. The ELF header comprises the file magic data, class type, program entry points, target application binary interface (ABI), file interpretation indicators, sizes, and addresses to program headers and section headers (Raju et al., 2021). Shahzad and Farooq (2012) performed the ELF headers empirical analysis of the malware and benign ELF files and confirmed that the some of the features can be used to distinguish the malware and benign ELF files. They used the ELF headers as a feature set to detect the IoT malware. The ML models applied on the selected features achieved more than 99% detection accuracy. However, the malware samples considered for this study are generic Linux malware covering various attacks and not particularly represent the IoT malware files. So, we may not be sure if the features selected for Linux malware can be applicable to detecting the IoT malware or not.

Strings: One of the definitive functionality supported by most of the IoT malware is Command and Control server connection for forming a IoT botnet to target the victims with DDoS attacks. There is highly likely that the remote C&C server and IP address can be seen in the printable strings of IoT malware binary. So, malware binary printable strings also seen to be used as an essential feature for IoT malware detection and classification. In Alhanahnah et al. (2018), the author's taken into account the printable strings from the ELF binary and extracted n-gram string vectors to cluster the samples and then used for IoT malware signature generation to classify the IoT malware. In addition to the string features, 8 high level statistic features like total number of functions, total number of call instructions are considered during the malware classification process in Alhanahnah et al. (2018).

Opcodes: The malware binary operational code (opcode) statistical features may be taken into consideration as a feature set for IoT malware detection and classification. The malware file is decompiled to extract the operation codes and utilized for applying in malware classification. The authors in HaddadPajouh et al. (2018) extracted the opcodes features for the ARM based malware and benignware using objdump tool. Then, Term Frequency-Inverse Document Frequency (TFIDF) is applied along with information gain as a feature selector to obtain the top 10 highest Opcode values present in the samples. Word embedding is applied to map the Opcode features into numerical sequence representation. Consequently, conventional ML and long short-term memory (LSTM) with layers varying from 1 to 3 are applied to classify the ARM based malware and benignware. LSTM model with 2 layers achieved the best performance accuracy 94% among the models used for evaluation. But, there is still a room to improve the performance of the ARM based malware classification and the number of malware samples taken for this study is very less (281). Darabian et al. (2020) studied the Opcode based polymorphic IoT malware detection using ML models. In specific, the maximal frequent patterns of opcode sequences are considered as a feature and applied k-nearest neighbor (KNN), Support vector machine (SVM), Multi-layer Perceptron (MLP), AdaBoost, Decision tree, and Random forest (RF) classifier models for malware detection. The authors reported that they achieved 99% accuracy for detecting the unseen malware. This work is limited to ARM based malware dataset study and malware classification of IoT malware supporting various CPU architectures is desired. In addition, the dataset preprocessing is required to extract the Opcodes from ELF binary files.

Control Flow Graph (CFG): The CFG of a binary file represents the order of execution of opcodes. It captures the relationship between the opcodes run sequentially when the corresponding ELF file is executed in a machine. Alasmary et al. (2019) presented a CFG based solution to detect and classify the IoT malware and also analyzed the differences between IoT malware and the android malware in terms of the CFG properties. The malware binaries are disassembled and extracted the CFG using radare2 automation process. The CFG features such as nodes, edges, density, the shortest path etc. are used as the dataset feature and applied SVM, logistic regression (LR), RF and CNN models for malware detection and classification. The results reported in the paper shows that CNN achieved the best performance with accuracy 99% for malware detection and classification. However, the 3 IoT malware families are considered for the IoT classification problem and the extraction of the CFG required preprocessing binary samples. This may be time-consuming and need additional efforts. Nghi Phu et al. (2021) proposed CFG based dynamic programming algorithm for fast extraction of the CFG based features from the disassembled malware code. The proposed algorithm is applied on MIPS and X86 based IoT malware dataset and used SVM for malware binary classifica-

tion. However, the IoT malware is operated on heterogeneous CPU architecture and not limited to MIPS and X86.

Image representation: The malware binary files are converted into an image with binary file raw bytes are mapped into the image pixels for malware detection and classification. The image based detection has been extensively used in windows based PE malware detection and the results obtained in the literature are promising (Nataraj et al., 2011; Venkatraman et al., 2019). However, the cross-compiler based IoT malware detection and classification can be challenging, as the image representation is the direct reflection of the malware binary file raw data and the different CPU architecture uses different instruction sets to store the program data. Furthermore, the malware to image conversion process may incur preprocessing delay and additional burden for real time malware detection and classification.

Dynamic Analysis: The IoT dataset feature extraction using dynamic analysis environment is more complex than setting up a typical virtual environment because of the various CPU architectures supported by the IoT malware. So, there is a limited work seen in the prior art utilizing the dynamic analysis features for IoT malware classification. Jeon et al. (2020) proposed a dynamic analysis based IoT malware detection, in which the dynamic behavior features such as memory, network, virtual file system, process, and system call are extracted; converted those features into an image and then applied CNN to perform IoT malware detection. The authors reported that their proposed method achieved 99.28% accuracy for a dataset sample size 1400.

Hybrid Analysis: Similar to dynamic analysis work in IoT malware space, there is a limited work has been done of utilizing the hybrid techniques for IoT malware detection. Baek et al. (2021) proposed two stage hybrid IoT malware detection using bidirectional- long short-term memory (Bi-LSTM) and EfficientNet-B3 model. In the first stage, the static feature opcodes are extracted from the ELF files and then passed through Bi-LSTM to filter the malware files. The benign detected files are analyzed in virtualization environment to extract the API calls, process memory and applied EfficientNet-B3 to detect the missed malware files in the first stage. The two stage hybrid model reported 94.46% performance accuracy for IoT malware detection. The performance can still be improved with new detection models.

Apparently, all the above-mentioned static analysis based feature extraction methods may require the preprocessing capabilities and needed additional set of tools for extracting the data sample representation features. Furthermore, most of the prior art works did not take into account of the heterogeneous CPU architectures supported by the IoT malware when using ML/DL methods for IoT malware detection and classification. Recently, the byte sequences from the entry point of the ELF binary file are used as static features for detection and classification of the IoT malware. The byte sequence from the entry point may provide correlation information among the same malware families and able to distinguish the malware from benignware as the malware variants usually takes the existing code bases and do the minimal changes like adding remote IP addresses, and domains for remote communication. We have leveraged the static feature byte sequences as a feature set for our DL based approach. The Table 1 describe the strategies used in the prior art research work and also the comparison of our byte sequence feature approach with the state-of-the-art techniques to improve the malware detection and classification accuracy. Further, We discuss the related work using byte sequences as feature in the literature for IoT malware detection and classification in detail.

2.1. Related work

Kumar and Lim (2019) converted the 9,000,000 byte sequence of the sample file into 300 byte sequence using Truncated Sin-

Table 1
Conceptual comparison of our work with related prior art work.

Authors	Device	Feature type	Strategy	Characteristics
D'Angelo et al. (2020)	Android	Dynamic	API-Images and Autoencoder	Android App API calls dynamic behavior information
Nguyen et al. (2018)	IoT	Static	PSI graph and CNN	ELF file PSI graphs to capture semantic information
Kumar and Lim (2019)	IoT	Static	Raw byte Sequence and RNN with LSTM	ELF file byte sequence features to eliminate the need for domain knowledge
Bakhshinejad and Hamzeh (2019)	IoT	Static	Byte Sequence and CNN	Byte sequences along with Word2vector model
Alasmary et al. (2019)	IoT and Andriod	Static	Graph based CFG and CNN	Graph related features from CFG
Dovom et al. (2019)	IoT	Static	Opcodes, fuzzy and fast fuzzy pattern tree	Opcodes along with fast fuzzy patterns for pattern recognition
Niu et al. (2019)	IoT	Static	OpCodes, Extreme Gradient Boosting (XGBoost)	Opcode feature fusion at three different levels
Phu et al. (2019)	IoT	Static	C500-CFG and SVM	C500-CFG to process large files and extract the features
Wan et al. (2020a)	IoT	Static	Byte Sequence, N-gram vector and SVM	The 4-gram byte seqeunce features
Proposed	IoT	Static	Byte Sequence,Bidirectional GRU and CNN	ELF file byte sequence with bidirectional byte processing

gular value composition (SVD) and then applied recurrent neural network (RNN) with LSTM on the 300 byte sequence as a feature length to detect if the given file is malware or not. If the file byte sequence length is less than 900000, then the file is padded with zero sequences. The results reported shows that the authors method achieved 91% on detecting the IoT malware. On the other hand, the authors also proposed CNN based shallow model with parallel convolution layers and converted the byte sequences as an image for feature set. This model achieved 99.1% accuracy for malware detection. Both of these models still required preprocessing steps to obtain the feature set and the byte sequence feature applied for LSTM model performance is nominal.

Wan et al. (2020a) proposed byte sequence from entry point of the file using KNN, SVM, Naive Bayes (NB), and MLP algorithms for IoT malware detection and classification. The byte sequence is represented as a vector form having the values of n-gram and the frequency of n-grams in the byte sequence. The byte sequence length has been varied from 128 to 1024 to find the optimal length and tested on a dataset containing 111k both malware and benign IoT samples each. The authors mentioned that the SVM performed slightly better than the other models with accuracy of 99.96% for IoT malware detection and 98.47% in IoT malware family classification. But, in general, ML models may not learn in incremental basis and real time learning capability may be low.

In Wan et al. (2020b), the authors proposed two feature extraction algorithms based on levenshtein distance and p-spectrum and applied ML algorithms KNN, SVM to detect and classify the IoT malware. levenshtein distance quantifies the difference between two input strings to represent as a feature. In p-spectrum, the byte sequences are converted into numerical vectors using the sequence substring “p”. The obtained results show that p-spectrum SVM (PS-SVM) performed better than other classifiers with accuracy 99.93% for IoT malware detection. Further, the classifier levenshtein distance based KNN (LD-KNN) performed good enough to achieve 99.24% accuracy. These methods require preprocessing efforts of the byte sequences prior to applying ML models.

Samantray and Tripathy (2021) leverage the first “N” byte sequence as a feature set and applied the ML algorithms RF, DT, NB, and Support Vector Classifier (SVC) to detect the given IoT binary is malware or not. The results reported that the SVC performed well compared to the other models with accuracy of 95.43%. But, the paper did not perform the malware classification based on the IoT malware families and also the performance still has a room to improve for better detection. All of these byte sequences based approaches used in the prior art has either require more preprocessing steps to obtain the expected feature sets or the methods applied for IoT detection and classification still has room for performance improvement.

To address these two issues, we have leveraged the byte sequence from the starting point of the binary file to eliminate the preprocessing steps. Additionally, our approach can be applied by anyone with limited malware domain knowledge. Furthermore, we have proposed DL based model Bi-GRU-CNN for effective, robust, platform independent IoT malware detection and classification after thorough performance analysis of the various DL models.

The Table 2 refers to the comparison of various static feature types and various ML and DL models used to obtain the performance of IoT malware detection and classification. The Table 2 is categorized based on the objective of the research, feature, dataset sample size, ML/DL model used and their performances for comparing our work with the prior art.

3. Proposed methodology

In this paper, we address the malware detection and family classification problem related to the IoT. The ML models along with feature extraction models such as static analysis, dynamic analysis, and malware image representation are being used recently to detect and classify the malware. But, these models require extensive feature engineering to extract the relevant features and security domain knowledge to select the important and meaningful features for effective malware detection and classification. Some of the feature extraction techniques like dynamic analysis in IoT is much more complex to set up the environment and capture the corresponding dynamic features, as the IoT malware run on diverse operating systems supported by heterogeneous CPU architectures. Recently, DL models are being proposed for malware detection and classification. The DL models application requires minimal domain knowledge and the DL models learns the input data features automatically with their structure and semantics memorizing capabilities. The existing ML/DL based solutions in IoT security highly rely on the static features of the malware samples (Alasmary et al., 2019; Dovom et al., 2019; Nghi Phu et al., 2021). But, most of the existing works has feature engineering problem issues or still has a scope to improve the performance accuracy. Additionally, the IoT malware has evolved over the time as the variants of IoT malware like Mirai, Hajime are seen to be exploited in the wild. So, precise IoT malware family classification is challenging because of the code reuse in the IoT malware. It is also imperative that any good malware detection mechanism clearly distinguish the malware and benignware without compromising on the time takes by the model to produce the result. In particular, the low resource enabled IoT devices should respond quickly to malware detection when the device is infected with malware. So, it is highly desired to not consume time for feature extraction/engineering.

To address the above-mentioned issues, we propose a byte sequence from the entry point of binary program as a feature approach for effective, robust, platform independent IoT malware detection and family classification using DL based model Bi-GRU-CNN. The byte sequences from the entry point of the malware are capable of distinguishing the structural and semantics of the malware families, since the source code when the program main function starts not usually changed in the same malware family variants. We believe that this unique feature enable to accurately distinguish malware and benignware and classify the malware family. But, we are aware that an adversary may use malware obfuscation techniques to move the entry point to another memory location in the binary file and the DL techniques may misinterpret the features to wrongly predict the malware detection and classification. Historically, the IoT malware did not extensively use obfuscation techniques and even the few percentages of malware identified as packed are packed with commonly known tools such as UPX. So, we tend to prefer the byte sequence as the ELF binary features for effective malware detection.

As shown in Fig. 1, the malware file is compiled with “gcc” to generate the compiled object file in ELF form. The ELF header contains the entry point address in the first 16 bytes of the file. The entry point value is referred as the memory address, where the program starts running when executed. From the entry point memory address, 2000 bytes are extracted and considered it as the byte sequences for that file sample.

3.1. Bi-GRU-CNN approach for IoT malware detection and IoT malware family classification

We have not considered advanced and complex deep models as currently the problem can be solved by using the deep learning algorithms. There are numerous algorithms available in Deep learning and mainly grouped into CNN and RNN. CNN generally applies on spatial data and RNN applied on sequential data. However, recent literature survey shows that CNN can perform better that RNN on text related problems in NLP and generally CNN was

Table 2
Comparison of our proposed work with related works for IoT malware detection and classification.

Authors	Year	Features	Objective	Byte Sequence Size	ML/DL Technqiues used	Sample size	Malware Families	Performance
Nguyen et al. (2018)	2018	CFG and PSI Graph	Detection	-	Deep Graph Convolutional Neural Network (DGCNN)	10,033	2	92%
Kumar and Lim (2019)	2019	Byte Sequence	Detection	300	RNN with LSTM and CNN	5572	2	91%(RNN with LSTM)
Bakhshinejad and Hamzeh (2019)	2019	Byte Sequence	Detection	1024	Parallel CNN	5560	2	97.1%
Alasmary et al. (2019)	2019	Graph based CFG	Detection	-	LR, SVM, RF, and CNN	6000	2	97.47(LR) 97.65(SVM) 98.48(RF) 99.66(CNN)
Alasmary et al. (2019)	2019	Graph based CFG	Classifier	-	LR, SVM, RF, and CNN	6000	3	97.22(LR) 97.23(SVM) 98.40(RF) 99.66(CNN)
Dovom et al. (2019)	2019	Opcodes	Detection	-	Fuzzy pattern tree	1207	2	99.83%
Niu et al. (2019)	2019	OpCodes	Detection	-	Extreme Gradient Boosting (XGBoost)	4169	2	94.5
Phu et al. (2019)	2019	C500-CFG	Detection	-	SVM	7000 MIPS	2	99.34
Wan et al. (2020a)	2020	Byte Sequence	Detection	128–1024	SVM,KNN,NB,MLP	222k	2	SVM 99.96%(MD)
Wan et al. (2020a)	2020	Byte Sequence	Classification	128–1024	SVM,KNN,NB,MLP	222k	8	SVM 98.47%
Wan et al. (2020b)	2020	Byte Sequence	Detection	64–1024	PS-SVM, PS-SVM, LD-SVM, LD-KNN	222k	2	99.93%(PS-SVM)
Wan et al. (2020b)	2020	Byte Sequence	Classification	64–1024	PS-SVM, PS-SVM, LD-SVM, LD-KNN	222k	7	99.96%(PS-SVM)
Nguyen et al. (2020)	2020	CFG and PSI Graph	Detection	-	CNN	11,200	2	98.7%
Hwang et al. (2020)	2020	strings from binary data	Detection	-	DNN	10,000	2	97.65%
Namavar Jahromi et al. (2020)	2020	Opcodes	detection	-	Two-hidden-layer Extreme Learning Machine (TELM)	552	2	99.65%
Samantray and Tripathy (2021)	2021	Byte Sequence	Detection	2KB	RF, DT, NB, SVC	36,328	2	95.43%(SVC)
Nghi Phu et al. (2021)	2021	CFG	Detection	-	SVM	5,476(MIPS)	2	99.19(MIPS)
Nghi Phu et al. (2021)	2021	CFG	Detection	-	SVM	6560 (Intel 80386)	2	99.06% (intel 80396)
Proposed	2021	Byte Sequence	Detection	2KB	RNN,GRU,BI-GRU,BI-RNN,BI-GRU-CNN	72,638	2	100% (BI-GRU-CNN)
Proposed	2021	Byte Sequence	Classification	2KB	RNN,GRU,BI-GRU,BI-RNN,BI-GRU-CNN	72,638	8	98% (BI-GRU-CNN)

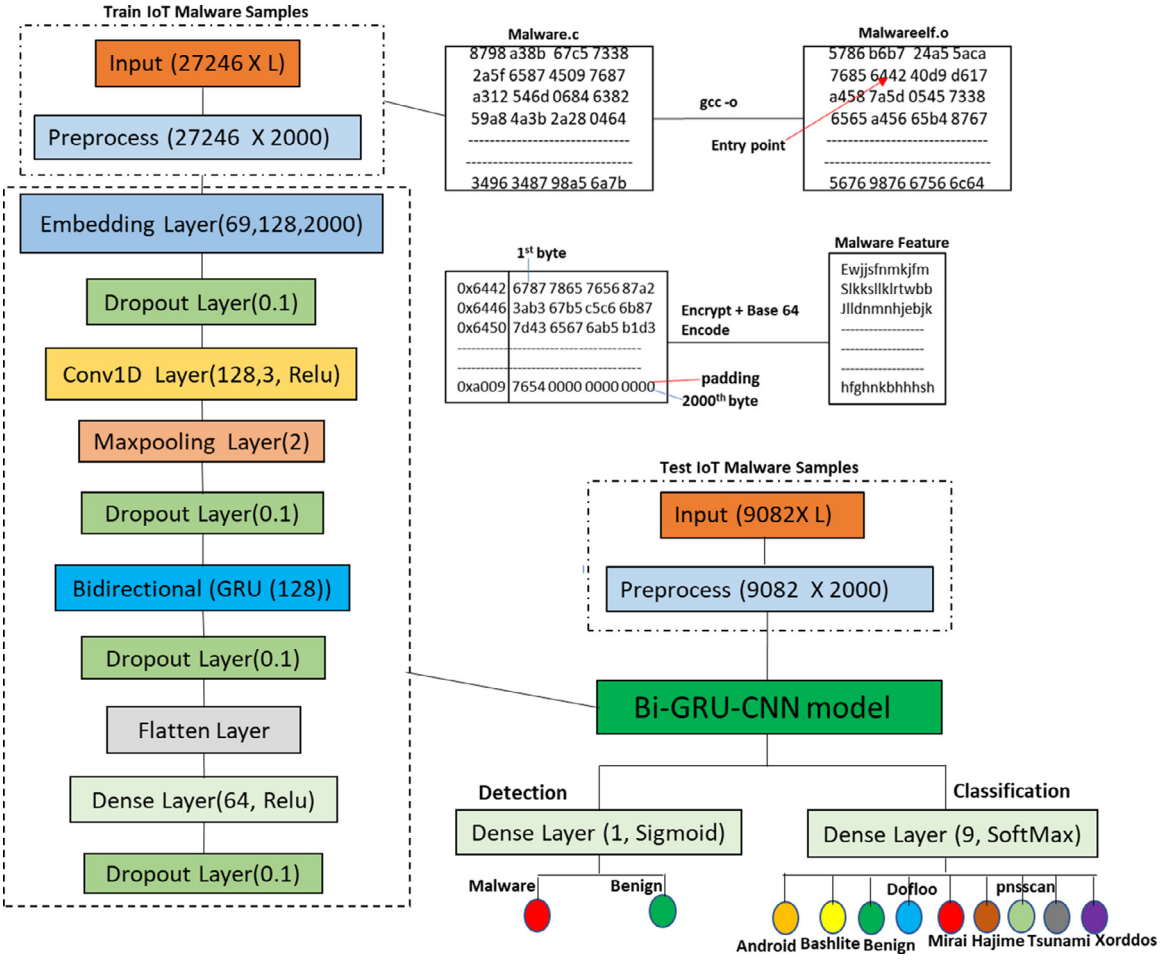

Fig. 1. Proposed model for IoT malware detection and classification.

well-known for image related problem. This indicates a detailed study is required to identify the algorithm that works better and our work contribution towards this direction for IoT malware application. The performances of these models depends on the network architecture and network parameters. In this work, we have done a detailed analysis of these models as this is not explored by others. Eventually, we have come up with a model that can be used for cross architecture IoT malware detection and classification. The proposed architecture include bidirectional GRU and CNN based deep learning approach.

The bidirectional GRU consist of two GRU's for processing the sequential data. One GRU takes the input sequence in forward direction and other GRU takes the input sequence in backward direction such a way that Bidirectional GRU can learn the previous and future features in the sequential data. The bidirectional GRU can be applied to data sequences for solving the Natural language processing problems. The CPU architectures reading and processing the code in either little endian or big endian format makes the bidirectional GRU is a potential application to use for platform independent malware classification. CNN are known to be used for image segmentation and classification problems. CNN can extract the temporal features and has the potential to perform for shorter text sequences. Due to the feature learning capabilities and ability to process the shorter text for classification, the CNN model is also integrated with Bidirectional GRU to solve the malware classification problem. Our approach of coming up with the Bi-GRU-CNN model for malware classification is discussed in the following paragraph.

Since the byte sequence represented IoT malware detection and classification problem is considered as NLP problem, the NLP preprocessing techniques are adopted and tested against various recurrent neural networks. To select the optimal DL model, we evaluate various DL models and as well as the bidirectional combinations of the DL models. Initially, the RNN, LSTM and GRU has been chosen and applied to the test dataset for IoT malware detection and family classification. To learn the byte sequence patterns from both directions of byte values, the bidirectional RNN, LSTM, and GRU are applied to the malware sample test set. The bidirectional models have an advantage of the producing better accuracy and reduce the number of parameters required to train the model. Furthermore, the best performed bidirectional GRU is integrated with CNN model to further improve the performance and obtain best results. The detail description of each layer used in the proposed Bi-GRU-CNN for IoT malware detection and classification is shown in Fig. 1.

Input Layer: This layer can have either malware or benignware samples for training and testing the samples in the input block. It comprises the set of the dataset samples having length varying between 0 and L as an input. The maximum length of byte sequence observed in the dataset malware or benignware samples was 2733. The preprocess block is part of the input layer and perform the length shortening, encryption, and encoding to represent the byte sequences in human-readable formats. When the maximum length 2733 in the IoT malware sample size is used for malware classification, the optimal classification performance is achieved. But, the training time is higher than the training time required when train-

ing the malware samples reduced length. As the file header values, program sections and CPU architecture information mainly seen in the first few byte sequences of the file for distinguishing the malware and benignware, the file input length is curtailed to 2000 for achieving the similar performance as file maximum length 2733 samples case and reducing the training time. So, the default value used for L is 2000 in our experiments. Notably, we have also considered CPU architecture type as an input feature along with byte sequence in our experiments to test the proposed model for platform dependent malware detection.

Embedding Layer: The embedding layer is responsible to convert each word in the byte sequence into the vector number representation. The vocabulary size for the input byte sequence is considered as the input dimension parameter passing through the embedding layer. The vocabulary size is determined by counting the letters, single digit numbers and the special symbols that made up the input byte sequence. The vocabulary size 69 is obtained from 26 alphabets, 10 digits and the remaining special symbols. The output dimension of the layer would be 128 after embedding to represent the 2000 input length of malware or benign file. Since we have leveraged Bi-GRU for IoT malware detection, the output dimension is doubled to 256 by going through the bidirectional layer

Dropout layer: We have used multiple dropout layers in the proposed model to reduce the overfitting problem. Each dropout layer drops 10% percent (0.1) of the output of the previous layer. Followed by the embedding layer, the dropout layer is applied for reducing the number of parameters a bit.

Convolution 1D layer: The convolution layer is the major building block of the CNN. The input matrix consisting of the binary file word embeddings in each row are passed through the convolution 1D layer and each row is convolved with kernel size 3. The number of convolution filters used is 128. 'Relu' is adopted as an activate function in this layer.

Maxpooling Layer: The maxpooling is applied to the convolution layer output to reduce the dimensionality. This can also help to reduce the number of parameters and overfitting problem. We have performed maxpool operation with stride of window size 2 to select the maximum value of the selected non-overlapping region. The overall process downsample the size of the input by 2. Followed by maxpooling layer, we drop 10% (0.1) of maxpool output layer to reduce the overfitting problem.

Bidirectional GRU Layer: After the maxpooling layer, a bidirectional GRU is applied to process the data in both the forward direction and reverse direction of the sequential data. As shown in Fig. 2, the gated recurrent unit is a neural network consist of the update and reset gate. The past sequence data can be taken into consideration to predict the present data values. The reset gate "r" is used for the short-term memory and the update gate "x" helps to provide the long-term memory in GRU. The activation "a" shown in the Fig. 2 is a linear interpolation between the previous activation a_{t-1} and the candidate activation $\tilde{a}$.

The bidirection GRU layer takes the binary file byte sequence words embedded as a vector numbers and processed the vectors through CNN layers as an input and passes through the GRU to iterate the vector representation of words. It also uses another set of data with reversed vector representation of word sequence to apply on the other GRU in Bi-GRU. The Fig. 3 illustrates the Bi-GRU layer leveraging the two GRU layers to process the input and produces the sequence context output features. We have considered 128 units in each GRU layer of Bi-GRU. This layer followed by another dropout layer with 0.1 to reduce the overfitting problem.

Flatten layer: The output of the Bi-GRU layer is passed through the flatten layer for converting the Bi-GRU output into one dimensional array. We have obtained 256 values as the output of the flatten layer in our proposed Bi-GRU model.

Dense layer: Followed by the flatten layer, the features further reduced to 64 using dense layer. The dense layer taken into account of all the values in the Flatten output, applied the "Relu" activation function and generates the output. We have further used dropout layer for regularization purpose.

The last dense layer of the proposed model activation function is adapted as per the expected outcome classification of the model. For malware detection or binary classification, a *Sigmoid* function is utilized and the output dimension value 2 is selected to predict the given file is malware or not. On the other hand, for multiclass malware family classification, a "Softmax" function is utilized and reduced the output values to 9, which include 8 malware families and 1 benignware as shown in Fig. 1.

The step by step IoT malware detection and multiclass malware classification are shown in Algorithm 1 and 2. The input parameters such as number of data samples, length of the byte sequence for each sample, embedding layer output dimension, the number of epochs, vocabulary size, kernel size, and filter value for CNN, GRU units, pool size, and dense final output 2 are given as an input to the malware detection or binary classification. The output is simply the detection of the sample as either malware or benignware. On the other hand, for IoT malware family classification, the same parameters are passed as an input except the final dense output as 9. The output for the malware family classification is labeled as Malware family 1 (MalwareF1),..., Malware family 8 (MalwareF8) and Benignware.

Algorithm 1: IoT malware detection using Bi-GRU-CNN algorithm.

Input: malwaresamples=m; bytelength=l;
dense_embedding_dimension=d; vocabulary_size=v;
epoch=e; denseunits=x; denseoutput1=du1;
denseoutput2=du2(2); gruunit=g; filter=f;
kernelsize=k; poolsize=b
Output: Labels {*Malware*, *Benignware*}
for $i \leftarrow 1$ **to** m **do**
  $Byteseq_{li} \leftarrow$ preprocessing(i)
end
for $epoch \leftarrow 1$ **to** e **do**
  for $j \leftarrow 1$ **to** m **do**
    $E_{l1},\ldots, E_{ld} \leftarrow$ Embedding($Byteseq_{lj}$,v,d,l) ;
    $D1_{l1},\ldots,D1_{ld} \leftarrow$ Dropout($E_{l1},\ldots,E_{ld}$, 0.1);
    $C1_{l1},\ldots,C1_{ld} \leftarrow$ Convolution1D($D_{l1},\ldots,D_{ld}$, f, k, "elu");
    $M_{l_b1},\ldots,M_{l_bd} \leftarrow$ Maxpooling1D($C1_{l1},\ldots,C1_{ld}$,b);
    $D2_{l_b1},\ldots,D2_{l_bd} \leftarrow$ Dropout($M_{l_b1},\ldots,M_{l_bd}$, 0.1);
    $G_1,\ldots,G_{2*g} \leftarrow$ Bidirectional(GRU($D2_{l_b1},\ldots,D2_{l_bd}$, g));
    $D3_1,\ldots,D3_{2*g} \leftarrow$ Dropout($G_1,\ldots,G_{2*g}$, 0.1);
    $F_1,\ldots,F_{2*g} \leftarrow$ Flattern($D3_1,\ldots,D3_{2*g}$);
    $De1_1,\ldots,De1_{du1} \leftarrow$ Dense($F_1,\ldots,F_{2*g}$, du1);
    $D4_1,\ldots,D4_{du1} \leftarrow$ Dropout($De1_1,\ldots,De1_{du1}$, 0.1);
    $De2_{du2} \leftarrow$ Dense($D4_1,\ldots,D4_{du1}$, 2, "sigmoid")
  end
  return Binary label du2(2)
end

Even though no significant contributions are done in terms of introducing new algorithms in the current work, there are many limitations reported by the current work. One such limitation is the current model is not efficient in handling imbalance data. As we know there are few malware families that has less number of data samples in real-time. To handle this type of data, a special learning capability should be given to the models. We are working on proposing a new cost-sensitive learning model that will be integrated to the proposed work. We are working on adding adversar-

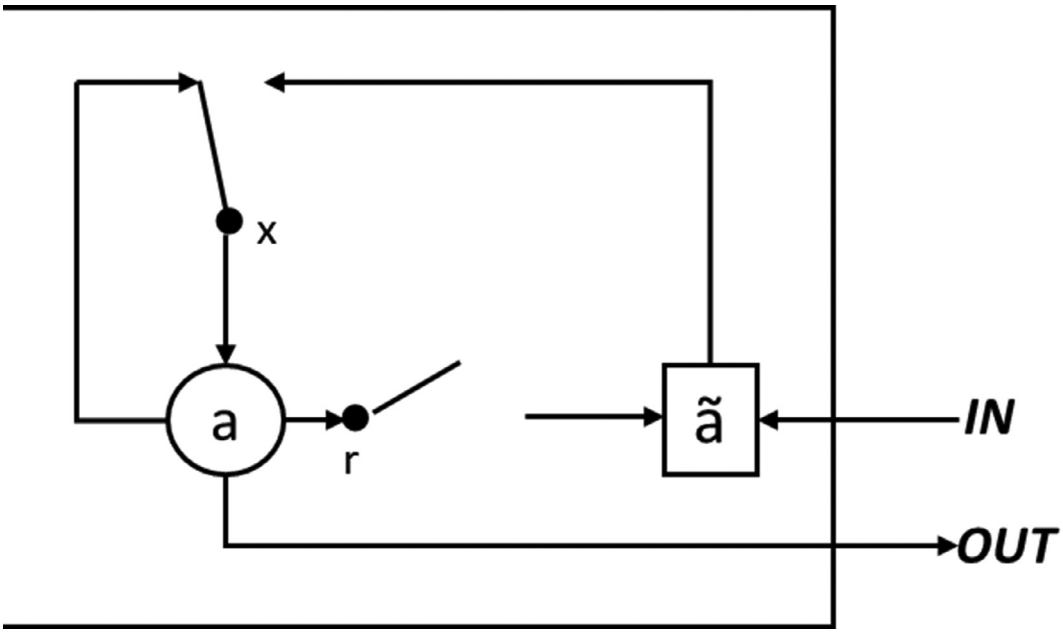

Fig. 2. Gated Recurrent Unit (GRU) "r" and "x" are the reset and update gates, and "a" and "$\tilde{a}$" are the activation and the candidate activation.

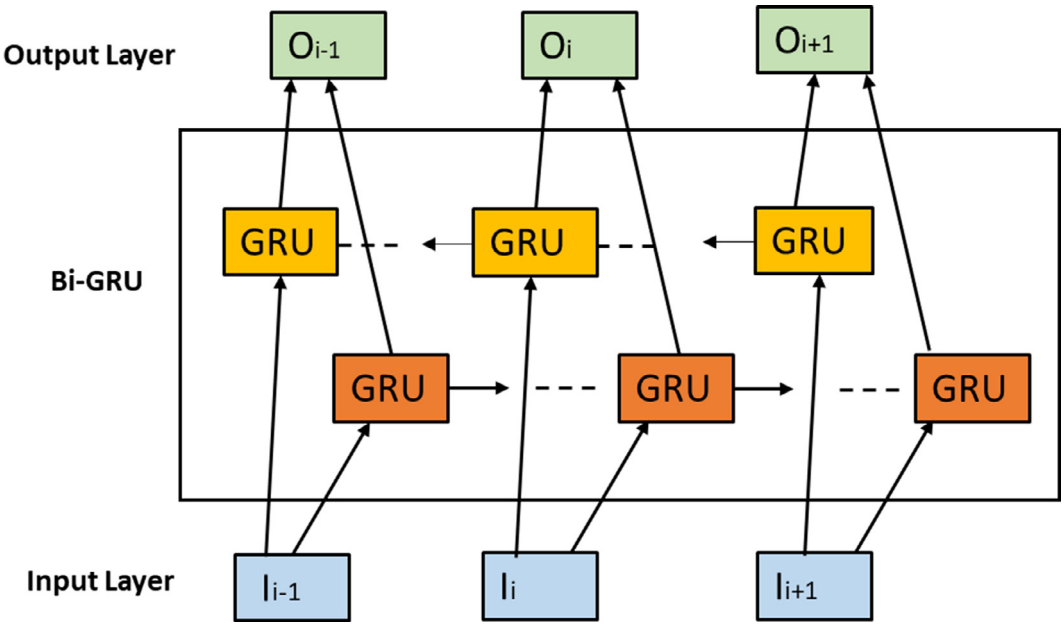

Fig. 3. Bidirectional Gated Recurrent Unit (GRU).

ial defense techniques to make the proposed model more robust against the existing adversarial approaches. Also, the future work list out the various possible way to bypass the proposed approach as well as prevent these attacks

4. Dataset description

As the IoT malware can target diverse IoT vendor products serving consumers with IoT application products, the IoT malware sample dataset may need to contain multiple malware families to study their behavior and classifying them. Since IoT devices operate on heterogeneous central processing units (CPU), IoT malware detection and classification requires much sophisticated solutions and rich malware dataset covering the ELF binary execution supported by different CPU. Our dataset contain 36,328 data samples, which includes both ELF malware and benignware collected from various sources. We have obtained this dataset from the existing resource for our evaluation (Center, 2021). Each sample in the dataset is represented with three columns. The first column represents the file disposition. The legitimate binary files are denoted as benignware and the files identified as IoT malware are denoted with their malware family to utilize for multiclass classification. The second column signifies the CPU architectures on which the sample is run and collected. The CPU families in the dataset include MIPS, ARM, X86, SuperH4 and PPC. The third column indicates the byte sequence from the entry point of the ELF binary program starts. The binary sequence extraction from the ELF file is as follows: The entry point address in the ELF header is identified and then the first 2k bytes from the entry point address are extracted. If the binary file length is less than 2k bytes, the byte sequence is

Algorithm 2: IoT malware classification using Bi-GRU-CNN algorithm.

Input: malwaresamples=m;bytelength=l;
dense_embedding_dimension=d; vocabulary_size=v;
epoch=e; denseunit=:x; denseoutput1=du1;
denseoutput2=du2(9); gruunit=g; filter=f;
kernelsize=k; poolsize=b
Output: Labels $\{MalwareF1, \ldots, MalwareF8, Benignware\}$
for $i \leftarrow 1$ **to** m **do**
  $Byteseq_{li} \leftarrow$ preprocessing(i) ;
end
for $epoch \leftarrow 1$ **to** e **do**
  for $j \leftarrow 1$ **to** m **do**
    E_{l1},...,$E_{ld} \leftarrow$ Embedding($Byteseq_{lj}$,v,d,l) ;
    $D1_{l1}$, ...,$D1_{ld} \leftarrow$ Dropout(E_{l1},...,E_{ld}, 0.1);
    $C1_{l1}, \ldots, C1_{ld} \leftarrow$ Convolution1D(D_{l1},...,D_{ld}, f, k, "relu");
    $M_{l/_b 1}, \ldots, M_{l/_b d} \leftarrow$ Maxpooling1D($C1_{l1}, \ldots, C1_{ld}$,b);
    $D2_{l/_b 1}$,...,$D2_{l/_b d} \leftarrow$ Dropout($M_{l/_b 1}, \ldots, M_{l/_b d}$, 0.1);
    G_1,...,$G_{2*g} \leftarrow$ Bidirectional(GRU($D2_{l/_b 1}$,...,$D2_{l/_b d}$, g));
    $D3_1$,...,$D3_{2*g} \leftarrow$ Dropout(G_1,...,G_{2*g}, 0.1);
    F_1,...,$F_{2*g} \leftarrow$ Flattern($D3_1$,...,$D3_{2*g}$);$De1_1$,...,$De1_{du1} \leftarrow$ Dense(F_1,...,F_{2*g}, du1);
    $D4_1$,...,$D4_{du1} \leftarrow$ Dropout($De1_1$,...,$De1_{du1}$, 0.1);
    $De2_{du2} \leftarrow$ Dense($D4_1$,...,$D4_{du2}$, 9, "softmax")
  end
  return Multiclass label du2(9)
end

padded with zero bytes for completion and maintain the consistent length among all the malware samples. The extracted 2k byte sequences in ASCII format are encoded with the encryption cipher to scramble the original ELF program content bytes. The encrypted content is fed to the base64 encoder to obtain Radix64 human-readable format data representation, which is added in the third column of the binary sample in the dataset.

Our selected feature extraction process doesn't require to run each sample in virtualized environment or perform deep reverse engineering using the existing tools like IDPro, Radare2, Objdump etc. The detail description of each malware family and their sample proportions in the malware are shown in the Table 3. The dataset comprises 8 IoT malware families Bashlite, Dofloo, Hajime, Mirai, Pnscan, Tsunami, Android and Xorddos and their corresponding sample count 7091, 19, 58, 8418, 4, 505, 247, and 11 in the dataset respectively. Additionally, the dataset contain 19,975 benignware samples taken from the legitimate binaries resided in the *\usr\bin* and *\usr\sbin* folders of Linux OS.

The IoT malware dataset samples are also categorized based on the supported CPU architecture. The Table 4 illustrates the different CPU architectures and their corresponding IoT sample count in the dataset. The MIPS based CPU architectures mipsel, mipseb, and mips64eb has the sample count 4981, 9052, and 1040 respectively. The ARM architecture "armel" has 9054 samples in the dataset. The X86 architectures X86el and x96_64el samples 3610 and 2752 are also included in the dataset. The dataset also contain 2345 PPC, 1276 Super6 and 1215 Sparc processor samples. The 1003 malware samples in the dataset are unknown CPU architecture type.

The main difference between the different CPU architectures is the supported CPU 32 bit or 64 bit, little endian or big endian system and the instruction set needed to support either Reduced instruction set computer (RISC) or Complex instruction set computer (CISC). These features can be leveraged to distinguish the CPU architectures supported by executable files. In our dataset, the ELF

Table 3
IoT malware family dataset description.

IoT Malware Family	Samples	Malware Description
Android	247	The android malware mostly infected through website advertisements or malicious apps in the app store on mobile phone.
Bashlite	7091	It exploits the shellshock vulnerability or brute forcing default credentials and can initiate multiple open network connection on the compromised devices to support DDoS attacks.
Benignware	19,975	The benignware Elf files are the benign Elf binaries taken from the various Linux operating system benign files.
Dofloo	19	Dofloo malware is known to be used for performing DDoS attacks, information stealing including the infection device information, device configuration.
Hajime	58	Hajime malware has similar characteristics of mirai botnet to compromise the poorly configured IoT devices. It uses P2P protocol to connect back with malware operator servers.
Mirai	8418	Mirai is known to be one of the familiar malware targeting the IoT devices. It supports multiple network protocol exploitation for reflection/amplification attacks and other DDoS attacks.
Pnscan	4	Pnscan malware used as a reconnaissance tool to find vulnerabilities in the infrastructure.
Tsunami	505	Tsunami malware family infect the machines using known vulnerabilities in the IoT devices to successfully compromise the IoT devices.
Xorddos	11	Xorddos uses the XOR encryption when communicating with Command&Control server and use SSH brute force to install the malware.

Table 4
IoT malware CPU family dataset description.

Sample count	CPU Family	CPU description
4981	mipsel	Mipsel is a little-endian MIPS architecture.
3610	X86el	X86 32-bit little-endian architecture
9052	mipseb	Mipseb is a big-endian MIPS architecture
2345	ppceb	PPC big endian architecture
1276	sh4el	superH4 little endian architecture
9054	armel	ARM little endian processor
1215	sparceb	Sparc big endian processor
1003	unknown	-
2752	X86_64el	X86 as well as 64-bit system supported little-endian architecture
1040	mips64eb	Mips64eb is a 64 bit little-endian MIPS architecture

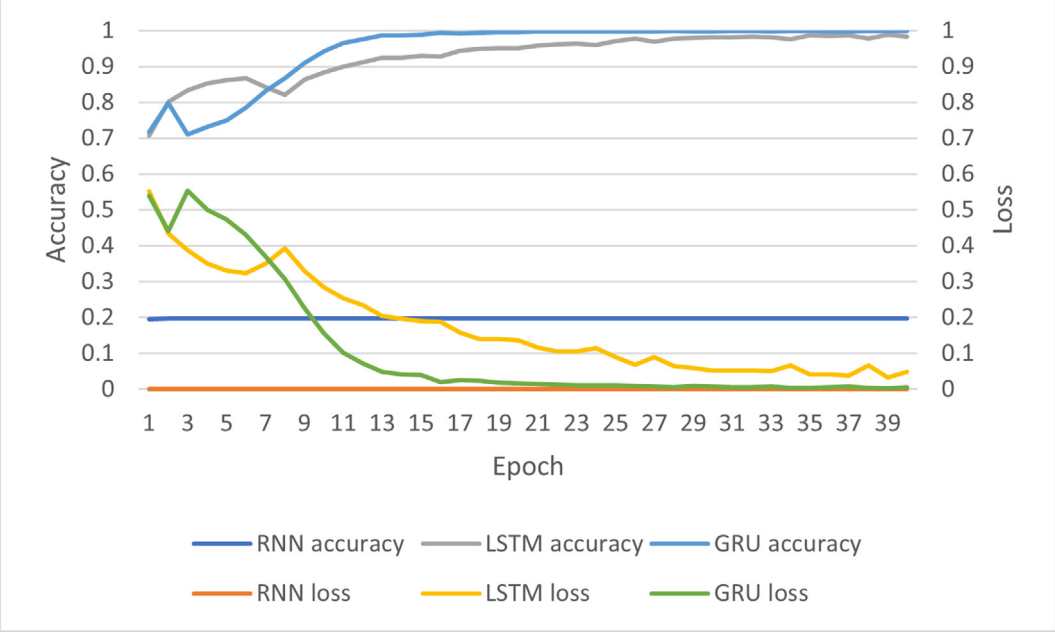

Fig. 4. Malware binary classification of RNN, GRU and LSTM models.

malware and benignware samples supporting various CPU architectures considered for evaluating the IOT malware detection. The ELF header stores the essential supported CPU architecture information such as 32 or 64 bit, little endian or big endian system that the executable can run. So, parsing the few first few bytes in elf file and processing through the NLP models is the key to distinguish the supported CPU architectures. So, in addition to the binary code and data variables stored in the section header of the ELF file, the ELF header plays a major role to distinguish the malware in cross-platform architecture malware detection. On the other hand, to identify the malware classes from the same CPU architecture, the section header byte values are the crucial to capture the binary code patterns by NLP and deep learning models to distinguish the malware. In this work, we rely on both the ELF header bytes and the section byte sequence to classify the malware irrespective of the running CPU platform. Hence, we have not considered the file full byte sequence for our detection and restricted to first 2k bytes.

5. Experimental results and discussion

In this section, the experimental evaluation of the byte sequence approach on the various deep learning model combinations along with the proposed model is performed and provided the detail description of the obtained performance results.

Firstly, The performance evaluation parameters of the multi class malware classification are described here. In order to describe the evaluation parameters, the confusion matrix parameters are defined as follows. True positive (TP) for a malware class is measured as the correct prediction of the malware class sample when the same malware class sample is actually present. The TP is the diagonal value of the confusion matrix when the actual malware classes are represented as rows and predicated malware classes are represented as columns. False positive (FP) for a malware class is defined as the sum of the all values in the corresponding row except the TP diagonal value. False Negative (FP) for a malware class is defined as the sum of the all values in the corresponding column except the TP diagonal value. True negative (TN) for a malware class is defined as the sum of all the values of the confusion matrix excluding that malware class's column and row.

The accuracy calculation of the malware multiclass classification and binary classification use the same formula. As our work is focused on the multi class malware classification, we defined the accuracy for each class classification. So, the accuracy for a malware class is defined as the sum of the true positives and true negatives for malware class k to the sum of the true positive, true negative, false positive, and false negative for malware k.

$$Accuracy_k = \frac{TP_k + TN_k}{TP_k + TN_k + FP_k + FN_k} \tag{1}$$

$$Precision_k = \frac{TP_k}{TP_k + FP_k} \tag{2}$$

$$Recall_k = \frac{TP_k}{TP_k + FN_k} \tag{3}$$

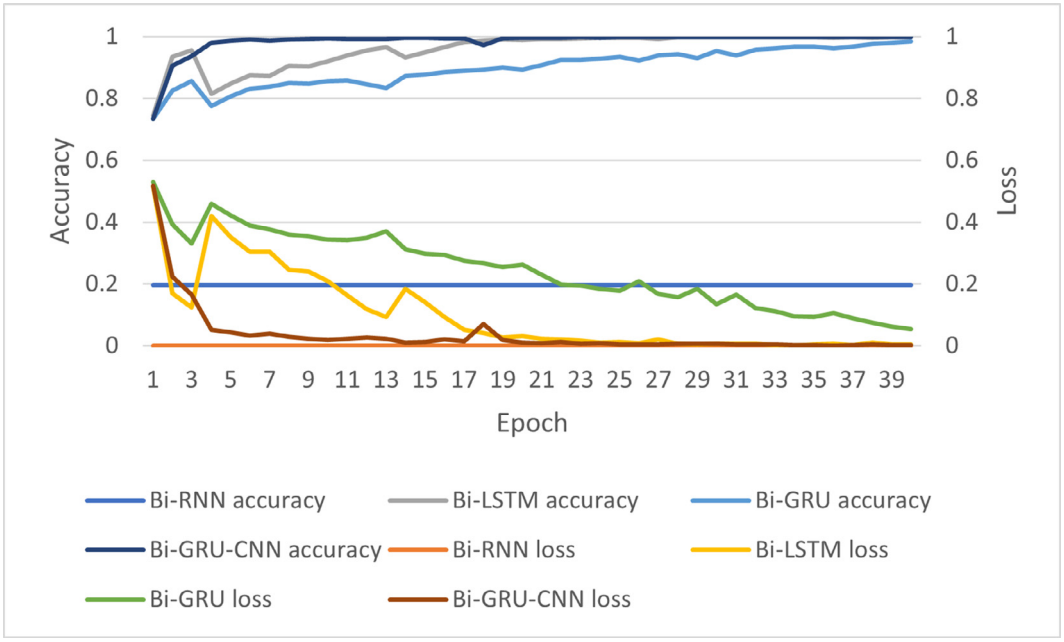

Fig. 5. Malware binary classification of Bi-RNN, Bi-GRU, Bi-LSTM, Bi-RNN-CNN models.

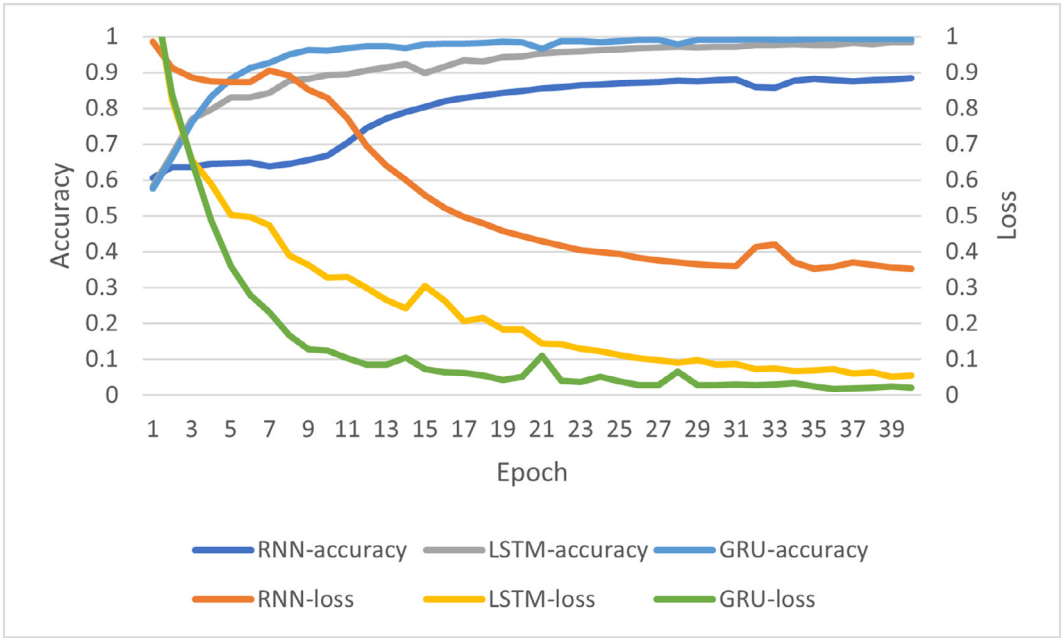

Fig. 6. Malware family classification of RNN, GRU, and LSTM models.

$$F1 - Score_k = \frac{2 * Recall_k * Precision_k}{Recall_k + Precision_k} \quad (4)$$

The precision for malware class k is defined as the ratio of the true positive for class k to the sum of true positive and false positive for class k. Similarly, the Recall for malware class k is determined as the ratio of the true positive for class k to the sum of true positive and false negative for class k. The higher the values of precision and recall of a model, the better the model is performed. F1-score of class k is the harmonic mean of precision and recall of class k.

The experimental setup consist of a system configuration of 16 GB RAM and Nvidia Telsa P100 running on Kaggle data science competition platform. The software packages Keras[1] with TensorFlow[2] backend, scikit-learn[3] were installed to write the code and obtain the performance metrics of our evaluation.

The IoT malware and benign dataset samples are split into training and testing samples with proportions 75 and 25. The loss functions *binary_crossentropy* and *categorical_crossentrophy* are used for malware detection or binary classification and malware family classification respectively. The malware detection and binary classification are interchangeably used in this paper. The *adam* optimizer is used for both classification methods. Additionally, the

[1] https://keras.io/
[2] https://www.tensorflow.org/
[3] https://scikit-learn.org/stable/

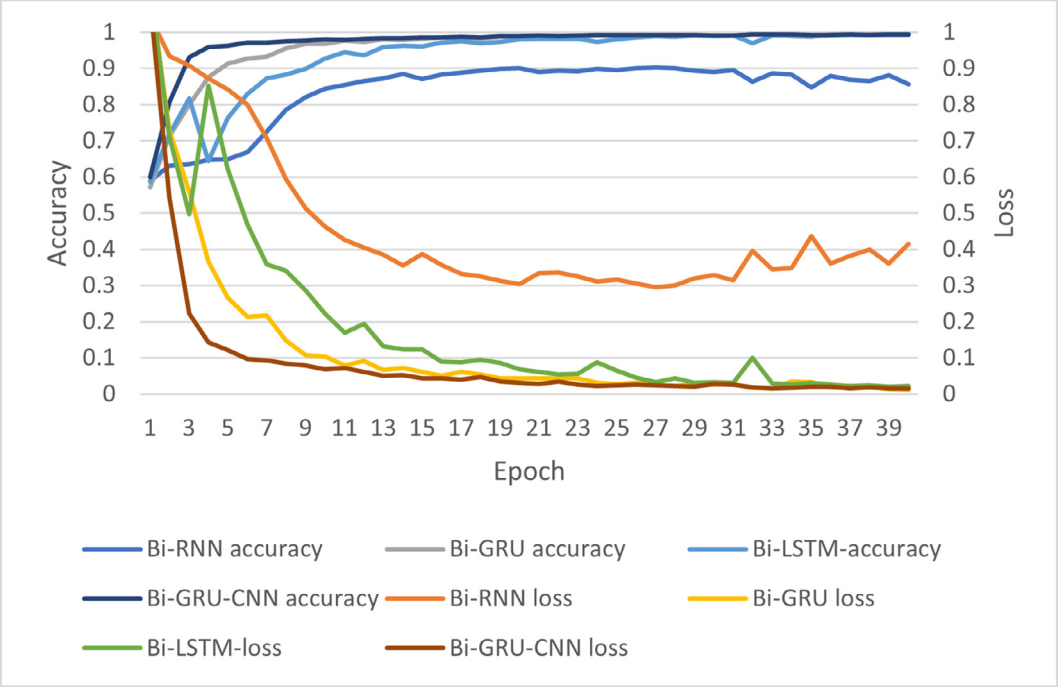

Fig. 7. Malware family classification of Bi-RNN, Bi-GRU, Bi-LSTM, Bi-RNN-CNN models.

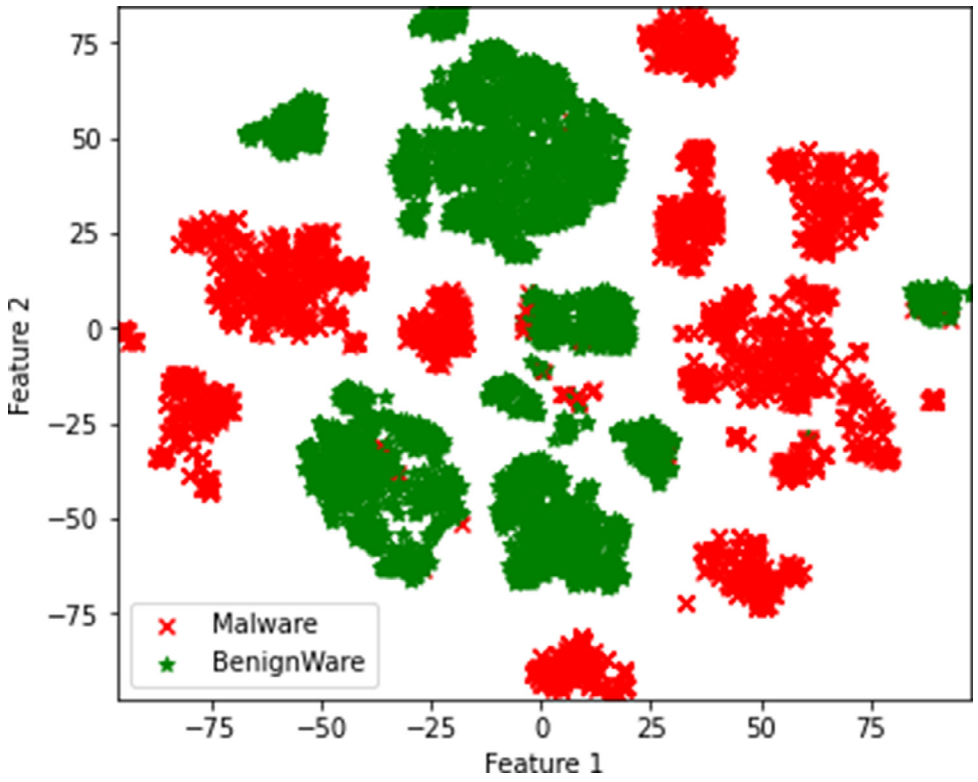

Fig. 8. t-SNE of GRU binary classification.

experimental parameters batch size, learning rate and epochs values 64, 0.01, and 40 are chosen to run the experiments for all the deep learning models used in our performance evaluation.

As our approach is based on the text representation of the IoT malware binary program byte sequences, the neural network models RNN, LSTM, GRU are considered for evaluating the malware classification performance. However, these model's current prediction values depend on the past data sequences. In order to incorporate the future data sequences to predict the current values, the Bidirectional RNN, LSTM, and GRU models are also considered in our evaluation models. As historically known, the combination of the CNN along with other models shown good performance in

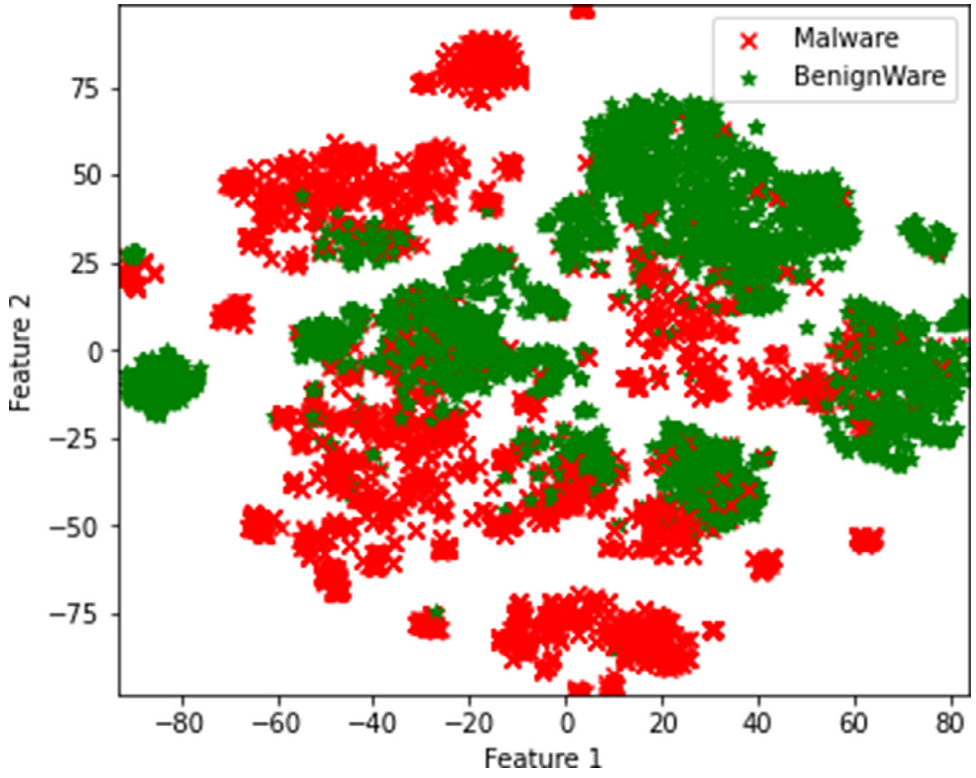

Fig. 9. t-SNE of Bi-GRU binary classification.

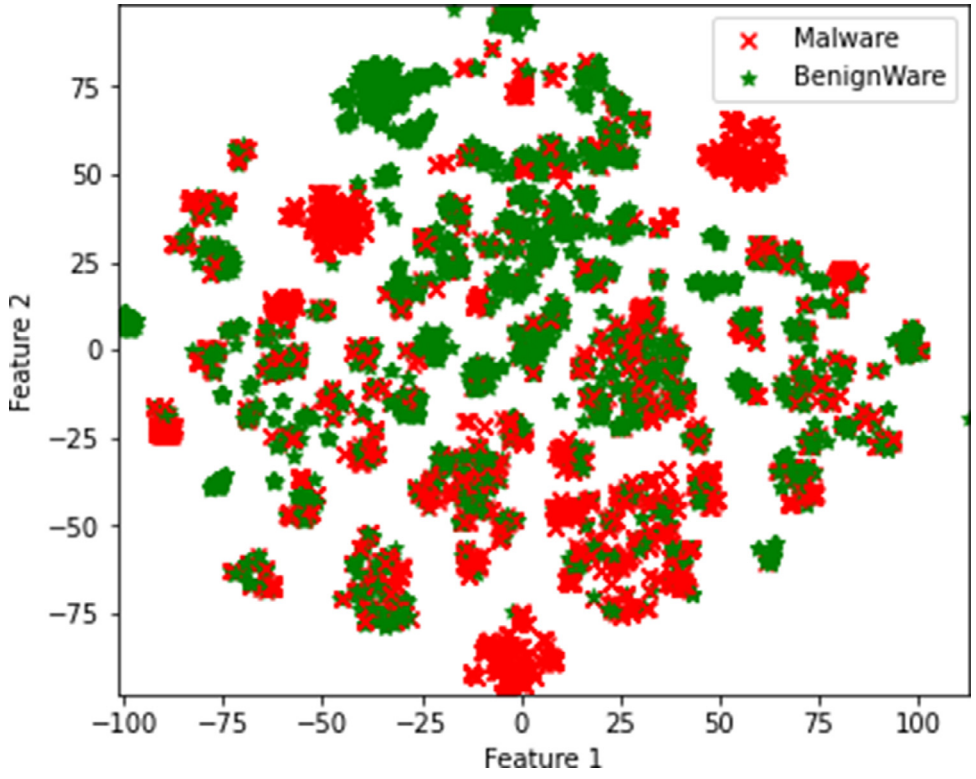

Fig. 10. t-SNE of Bi-GRU-CNN binary classification.

Table 5
The Bi-GRU-CNN performance results when only byte sequence is considered as input feature.

Model	Class	Precision	Recall	F1-score
	Binary			
BI-GRU-CNN	Benign	1.00	1.00	1.00
	Malware	1.00	1.00	1.00
	Multiclass			
BI-GRU-CNN	Android	0.97	0.99	0.98
	Bashlite	0.96	0.97	0.96
	Benign	1.00	1.00	1.00
	Dofloo	0.67	0.25	0.36
	Hajime	0.87	0.93	0.90
	Mirai	0.97	0.96	0.97
	Pnscan	0.00	0.00	0.00
	Tsunami	0.76	0.77	0.77
	Xorddos	0.33	1.00	0.50

Table 6
Classwise performance comparison of malware binary classification(0:Benignware, 1:Malware).

Model	Class	Precision	Recall	F1-score
		Binary		
RNN	0	0.80	0.54	0.64
	1	0.70	0.89	0.78
LSTM	0	0.95	0.93	0.94
	1	0.94	0.96	0.95
GRU	0	1.00	1.00	1.00
	1	1.00	1.00	1.00
Bi-RNN	0	0.00	0.00	0.00
	1	0.54	1.00	0.70
Bi-LSTM	0	0.99	0.99	0.99
	1	1.00	1.00	1.00
Bi-GRU	0	0.95	0.95	0.95
	1	0.96	0.96	0.96
BI-GRU-CNN	0	1.00	0.99	1.00
	1	1.00	1.00	1.00

certain data types. So, we have selected the Bi-GRU-CNN for further performance evaluation and present the optimal performance model for malware binary and multiclass classification.

We have initially performed the malware binary classification using RNN, LSTM, and GRU. The input feature byte sequence and the supported CPU type are considered as the input. The Fig. 4 illustrates the training accuracy and loss for the three models RNN, LSTM, and GRU when the experiments run for epochs 1 to 40. Since the RNN does not remember the previous sequence data of trained datasets, the training accuracy is constantly maintained to be 20% for the RNN model. Consequently, the training loss for the RNN model is reported to be approximately zero. Based on these results, it is clear that the basic RNN model is not viable for byte sequence based malware classification. The LSTM model training accuracy has significantly improved as the epoch run from 1 to 25 and then settle down at approximately 97% by the end of epoch 40. Conversely, the training loss drastically reduced for the LSTM model when the epoch reaches at 28 and then maintained constant training loss with slight variations until the end of the epoch 40. The GRU achieved 100 percent accuracy when the epoch reaches almost 15, even though the model showed sluggish start during the initial epoch runs. From the epoch 15, the GRU model consistently maintained the 100 percent accuracy until the epoch 40. This shows that the GRU model quite significantly predict the malware classification outcome in the training dataset. Further, we can see that the training loss follows the opposite trend of the accuracy for the GRU model. The training loss showed downward trend starting from epoch 1 and almost showed zero training loss when the epoch reaches to 20. Then the GRU training loss consistently maintained to be zero until the end of epoch 40. Overall,

Table 7
Classwise performance comparison of DL models for malware family classification.

Model	Class	Precision	Recall	F1-score
RNN	Android	0.53	0.37	0.44
	Bashlite	0.88	0.81	0.85
	Benignware	0.86	0.93	0.89
	Dofloo	0.00	0.00	0.00
	Hajime	0.00	0.00	0.00
	Mirai	0.82	0.78	0.80
	pnscan	0.00	0.00	0.00
	Tsunami	0.68	0.22	0.33
	Xorddos	1.00	1.00	1.00
LSTM	Android	0.89	0.72	0.79
	Bashlite	0.90	0.92	0.91
	Benignware	0.93	0.93	0.93
	Dofloo	1.00	0.12	0.22
	Hajime	0.00	0.00	0.00
	Mirai	0.88	0.88	0.88
	Pnscan	0.00	0.00	0.00
	Tsunami	0.65	0.65	0.65
	Xorddos	0.00	0.00	0.00
GRU	Android	0.98	0.88	0.93
	Bashlite	0.96	0.95	0.95
	Benignware	0.99	1.00	1.00
	Dofloo	1.00	0.25	0.40
	Hajime	0.87	0.93	0.90
	Mirai	0.96	0.97	0.97
	Pnscan	0.00	0.00	0.00
	Tsunami	0.72	0.71	0.72
	Xorddos	1.00	1.00	1.00
BI-RNN	Android	1.00	0.97	0.98
	Bashlite	0.89	0.82	0.85
	Benignware	0.85	0.96	0.90
	Dofloo	0.00	0.00	0.00
	Hajime	0.93	0.93	0.93
	Mirai	0.89	0.74	0.81
	pnscan	0.00	0.00	0.00
	Tsunami	0.64	0.15	0.24
	Xorddos	0.50	1.00	0.67
BI-LSTM	Android	0.98	0.97	0.98
	Bashlite	0.95	0.94	0.95
	Benignware	0.99	0.99	0.99
	Dofloo	1.00	0.25	0.40
	Hajime	1.00	0.93	0.96
	Mirai	0.95	0.95	0.95
	Pnscan	0.00	0.00	0.00
	Tsunami	0.80	0.69	0.74
	Xorddos	0.00	0.00	0.00
BI-GRU	Android	1.00	0.99	0.99
	Bashlite	0.95	0.96	0.95
	Benignware	1.00	1.00	1.00
	Dofloo	1.00	0.25	0.40
	Hajime	0.93	0.93	0.93
	Mirai	0.97	0.97	0.97
	Pnscan	0.00	0.00	0.00
	Tsunami	0.86	0.63	0.73
	Xorddos	0.50	1.00	0.67

Table 8
Classwise performance comparison of Bi-GRU-CNN for malware family classification.

Model	Class	Precision	Recall	F1-score
BI-GRU-CNN	Android	0.99	0.99	0.99
	Bashlite	0.97	0.95	0.96
	Benignware	1.00	1.00	1.00
	Dofloo	1.00	0.50	0.67
	Hajime	0.93	0.93	0.93
	Mirai	0.97	0.98	0.97
	Pnscan	0.00	0.00	0.00
	Tsunami	0.86	0.78	0.82
	Xorddos	1.00	1.00	1.00

Table 9
Confusion matrix results for the GRU models malware binary classification case.

GRU		
	Benignware	Malware
Benignware	4136	12
Malware	15	4919
Bi-GRU		
	Benignware	Malware
Benignware	3952	196
Malware	197	4737
Bi-GRU-CNN		
	Benignware	Malware
Benignware	4127	21
Malware	15	4919

these results show that the GRU can comprehensively remember the byte sequences of the dataset samples for accurate malware binary classification.

RNN achieved the lowest performance with 20% accuracy because the length of the sequence is 2000 in our work and RNN is well-known to achieve better performance for small sequence data related problems. Since it doesn't have any additional memory and gating functions, the performance drastically reduced and also results in vanishing and error gradient issue. Literature survey shows that this was a long-standing problem and there were many directions of research was carried out by researchers to solve this issue. LSTM was introduced that includes a memory and gating functions to have a control on the memory like whether to store the information or discard while doing computation across the sequence. This solves the vanishing and error gradient issue and able to achieve better performance on various long-standing sequence based problems in NLP and speech processing (Lecun et al., 2015). As shown in Fig. 4, we clearly see that LSTM performed much better than RNN for this reason.

The Fig. 5 depicts the training accuracy and loss performance of the bidirectional neural network models Bi-RNN, Bi-LSTM, Bi-GRU and Bi-GRU-CNN. As seen in the RNN performance in the Fig. 4, the Bi-RNN also display the similar performance characteristics. The training accuracy for the Bi-RNN is 20% and this accuracy is constantly maintained throughout all the epoch runs. The training loss for the Bi-RNN is zero and in fact obtained the negative values, which are not shown in the Fig. 5. The Bi-LSTM training accuracy improved compared to the LSTM model for malware binary classification. The training accuracy showed upward trend starting from epoch 5 and obtained 100% training accuracy when the epoch reaches 19. Further, the 100% accuracy is maintained throughout the second half of the epoch execution. The Bi-GRU showed the upward trend starting from epoch 1 to 40, even though the model takes 40 epochs to achieve the 100% performance accuracy. On the contrary, the training loss showed the downward trend starting from the epoch 1. The model reaches almost zero loss when the epoch reached 21 and then settled down to zero until the epoch 40. The proposed model Bi-GRU-RNN clearly showed better training performance in terms of obtaining 100% training accuracy by the end of epoch 5 compared to the other model Bi-LSTM, which obtained the 100% by the end of epoch 20. The training loss for Bi-GRU-CNN also reached near zero within the first few epochs. These results portray that the proposed model Bi-GRU-CNN is well-trained to absorb the malware sequence characteristics and clearly distinguish the malware and benign file samples within few epochs in comparison with other bidirectional models.

The Figs. 6 and 7 illustrates the IoT malware family classification training performance accuracy and loss for the RNN, LSTM, GRU and the bidirectional models Bi-RNN, Bi-LSTM, Bi-GRU, and Bi-RNN-CNN. In contrast to the RNN model performance in binary classification, the RNN started off with 60% accuracy and constantly increased the training accuracy as the epoch changes from 1 to 40 and finally settle at accuracy 90%. Conversely, the RNN model training loss considerably decreased as the epoch reaches 40. But, the loss remained at around 0.35% by the end of epochs, which is substantial. The LSTM model training accuracy showed the upward trend and able to obtain 100% accuracy when the epoch reaches 35 and then remained steady till the experiment completed. The GRU model clearly showed the sharp increase in accuracy during the initial epochs and able to manage achieving 100% training accuracy within the 20 epochs. The training loss for the LSTM and GRU decreased as the epoch increases except few minor fluctuations in some epochs. But, it clearly illustrates that GRU produced steep downward curve with drastic loss reduction compared to LSTM loss performance. The bidirectional RNN model almost followed the similar training accuracy and loss patterns as of RNN model. On the other hand, the Bi-LSTM, Bi-GRU, and Bi-GRU-CNN models obtained the maximum training accuracy when the number of epochs reaches to 20 and constantly maintained accuracy until the epoch reaches 40. However, there is a clear difference between these three models on how quickly the models learn the byte sequences of the dataset samples. The best performed model Bi-GRU-CNN quickly races towards the maximum accuracy within the epoch 9, followed by Bi-GRU almost around the same epoch run and then Bi-LSTM reach the same level when the epoch is 21.

The training samples performance evaluation earlier clearly indicates that the combination of Bi-GRU with CNN achieved the best performance results when the input features include CPU type and byte sequences. In order to evaluate the best performed model Bi-GRU-CNN performance for platform independent case, we have performed the experiments with input feature as byte sequences alone. The Table 5 describes the binary and malware family classification of the Bi-GRU-CNN performance results in terms of the precision, recall, and F1-score. We can see that the Bi-GRU-CNN model correctly classify all the malware as well as benign files. This is an indication that our model can perform well with byte sequences even in platform independent scenarios. The malware family classification using the Bi-GRU-CNN shown promising results except the fact that the few malware families like pnscan, dofloo, tsunami, and xorddos obtained poor performance due to the imbalanced datasets and very few number of data samples exist in the test dataset for these malware families. The most frequently seen malware families like mirai, bashlite achieved precision of 0.97 and 0.96, which is noteworthy. Obtaining the balanced IoT malware dataset with IoT families may even produce good results using the proposed model Bi-GRU-CNN. This clearly shows that the CPU type as a feature has minimal impact on the IoT malware classification because the byte sequences incorporate the CPU type associated fields such as 32 or 64 bit architecture and little endian or big endian in the malware sample file ELF header, and the proposed approach could use these byte sequences to distinguish the malware classes to achieve similar performance.

We also present the class wise performance evaluation results of all the DL models considered in our evaluation to compare with our proposed model. The Table 6 illustrates the performance metrics of the DL models for IoT malware binary classification or malware detection. The performance metrics precision, recall, and F1-score are presented for comparison of these models. The RNN model and Bi-RNN models are least performed among the different models considered for our evaluation. The number of false positives and false negatives for the class of benignware and malware are much higher in the RNN case and interestingly Bi-RNN not able to classify the benignware correctly or incorrectly leads precision and recall value is zero. The LSTM model has performed nominally and achieved decent precision and recall, which is more than

Table 10
Confusion matrix for the proposed Bi-GRN-CNN model IoT malware family classification.

	Android	Bashlite	Benignware	Dofloo	Hajime	Mirai	Pnscan	Tsunami	Xorddos
Android	66	0	1	0	0	0	0	0	0
Bashlite	0	1673	3	0	1	64	0	11	0
Benignware	1	1	4929	0	0	3	0	0	0
Dofloo	0	2	2	4	0	0	0	0	0
Hajime	0	0	1	0	13	0	0	0	0
Mirai	0	31	6	0	0	2139	0	5	0
Pnscan	0	0	1	0	0	0	0	0	0
Tsunami	0	17	0	0	0	10	0	97	0
Xorddos	0	0	0	0	0	0	0	0	1

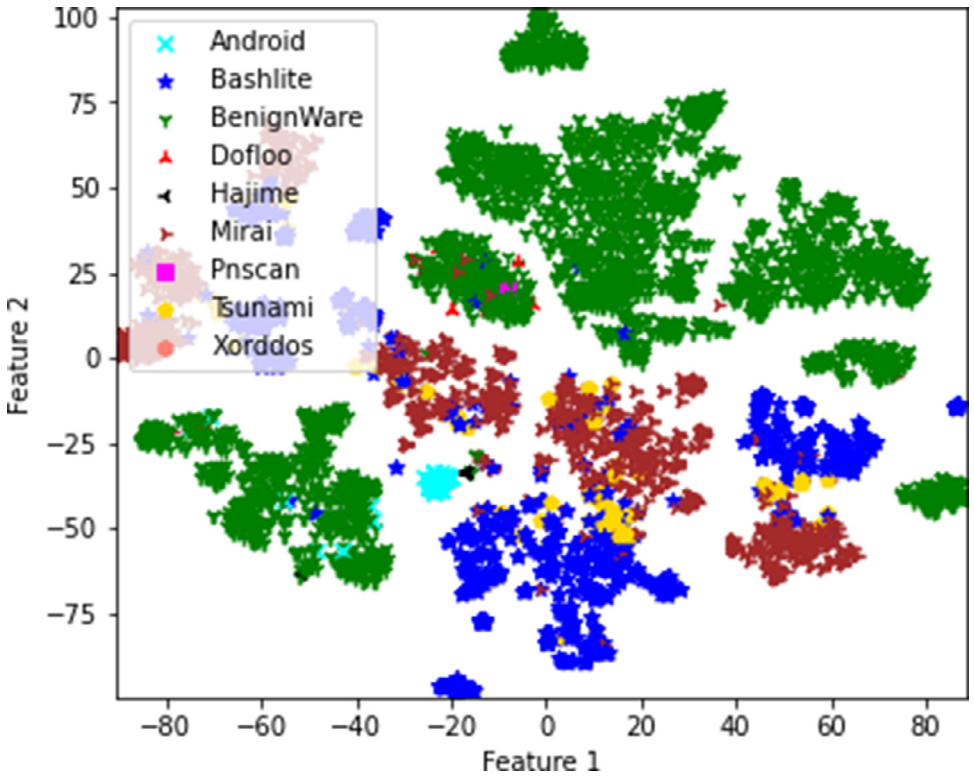

Fig. 11. t-SNE of GRU multi classification.

93%. Notably, even though GRU requires less number of features and also the time takes to process the input is less compared to the LSTM model, the GRU still achieved the best performance of all the models with no false positive and false negatives. The bidirectional model LSTM and GRU also performed well for malware binary classification. However, the Bi-LSTM obtained better precision and recall compared to the Bi-GRU, which is in contrast to the unidirectional DL models performances. When the Bi-GRU is combined with CNN model, the precision improved from 0.95 to 1 and the recall improved from 0.96 to 0.99 for benignware class performance. The same trend followed for the malware classwise performance as well when CNN is added to the BI-GRU model. Overall, these results showed that the GRU model performs well to process the byte sequences of the dataset binary files for malware detection. However, most of the DL models have misclassified the malware detection case and a detailed investigation and study of DL models is required to avoid the binary misclassification or missing malware detection.

The Tables 7, 8 illustrates the IoT malware family classification classwise performance results for the models used in our study. We can notice that some classes has zero misclassification results in the DL model results. This may be the fact that the dataset used for our study is imbalanced. As most of the IoT malware appeared in the wild are the variants of the few well known malware like bashlite and mirai families, balanced malware IoT dataset may be difficult to obtain from the public malware repositories. But, we may propose and apply new models to handle the imbalanced datasets. As shown in Tables 7, 8, hajime, pnscan, and xorddos families were not able to classified in some models. This may be due to the samples count for these families are very small. The dataset used for our study contain 58 samples of hajime malware family, 4 samples of pnscan and 11 samples of xorddos, which are too less compared to the total dataset 36728. Based on the results shown in Table 7, it is evident that GRU is performed slightly better than the LSTM and much better than the least performed model RNN for all the malware class family classification. The GRU model has achieved 99% precision and 100% recall for the benignware class and the highest sample in the malware family mirai class in the dataset also achieved good results for GRU.

In the bidirectional models, the same performance pattern is followed as the unidirectional DL models. The Bi-GRU slightly performed better than Bi-LSTM for most of the malware class families except the hajime and tsunami malware. The Bi-GRU-CNN model clearly outperformed than all the other models for all the malware families classification. The best performed Bi-GRU-CNN F1-score values for the top malware families mirai, bashlite, and Tsunami

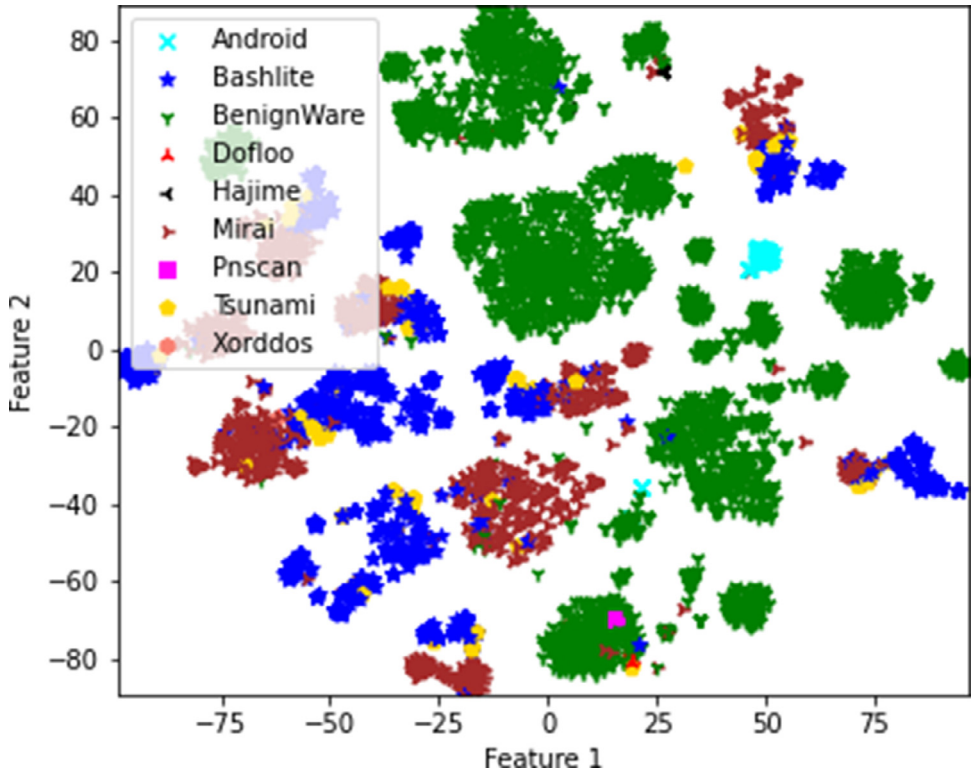

Fig. 12. t-SNE of Bi-GRU multi classification.

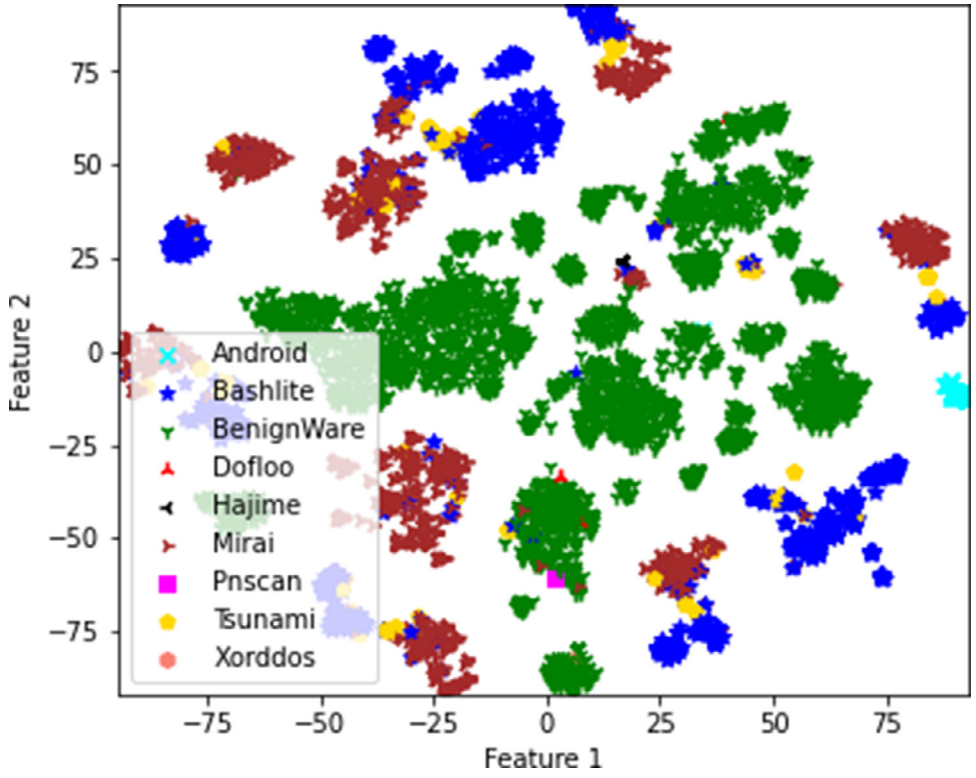

Fig. 13. t-SNE of Bi-GRU-CNN multi classification.

are as follows 0.97, 0.96, and 0.82 respectively. The misclassification possibility in some of the malware families can be because of using the same code excerpt to achieve the malicious tasks like perform bot connection to C&C server and brute force functionalities commonly seen in IoT malware.

To better understand the GRU model's performance in IoT malware detection and malware family classification, we present the confusion matrix results for the three models such as GRU, Bi-GRU, and Bi-GRU-CNN as shown in Tables 9 and 10. The GRU model is able to classify 4136 benignware samples correctly as benignware and 12 samples are misclassified as malware. On the other hand, 4919 malware samples are correctly classified as malware. But, 15 samples are misclassified as benignware. Interestingly, GRU performed better than the Bi-GRU. The Bi-GRU only able to classify 3952 benignware and 4737 malware samples correctly. But, overall, 393 samples are misclassified by the Bi-GRU, which is substantial. This performance reduction in Bi-GRU could be the result of using the byte sequences from starting point as well as ending point makes the input data more redundant to remember specific characteristics of the samples for malware classification. Finally, our proposed model Bi-GRU-CNN improved detection performance better than the Bi-GRU model. The Bi-GRU-CNN performance is comparable to GRU model classification. These results clearly show that GRU and the proposed Bi-GRU-CNN achieved similar results for the IoT malware binary classification. But, these two models did not follow the same trend for the malware family classification. GRU is known to be performing better when the sequence data is feed as an input data. However, it is limited to learn the spatial features. So, the consideration of the CNN models is a good choice for obtaining the fruitful results when spatial data is taken is an input. Furthermore, the bidirectional GRU process the sequential data in both the forwarding direction s well as reverse direction for better understanding the context and eliminating ambiguity, particularly, when the cross-platform supports both little endian and big endian systems. We can also clearly see that from Table 8, the proposed Bi-GRU-CNN outperformed the GRU model for multi class IoT malware classification.

We have discussed earlier that Bi-GRU-CNN performed better than other models for IoT malware families classification class wise performance comparison of all the models. The confusion matrix for the Bi-GRU-CNN model to classify the malware family classification is showed in the Table 10. We can see that the proposed model accurately classify majority of the malware families and benignware. However, some malware families like mirai, bashlite, tsunami falsely classify as the sample file belongs to other family. For instance, the 64 bashlite files are misclassified as mirai malware. The reason could be the fact that parts of mirai source code adopted from bashlite. Additionally, most of the IoT malware functionality and their intent would be the same. For instance, creating network connection to connect to the remote C&C server, functionality to initiate the DDoS attack by generating layer 3 or layer 4 attack packets are commonly seen in IoT malware. The above explanation is also true for the 31 mirai malware samples classified as Bashlite. So, the proposed method need to be studied further to identify this behavior and propose possible improvements for accurate detection.

5.1. Network parameters comparison

As the main objective of our study is to measure the detection accuracy of the proposed deep learning model to effectively detect and classify the cross architecture IoT malware, the training time and computational overhead is not monitored during training. But, the total network parameters, network layer wise details, and network structure details are compared for model training to relatively compare the computation overhead and training time for all the models. The training time is directly proportional to the number of parameters required to train the model. The computational overhead is also directly proportional to the number of parameters required to train the model. Table 11 shows the network parameters required to all the deep learning models for IoT malware multiclass classification. It is clearly evident that the Bi-GRU-CNN required slightly more network parameters compared to Bi-GRU model. But, the Bi-GRU-CNN model training time still be lesser than Bi-LSTM, as the B-LSTM model require more network parameters to train the model. In terms of network structure, network layer parameters, all the models have similar parameters such as number of units, embedding layer, dense, dropout layers and activation function. The same network structure and layer parameters used in the malware binary classification models except the binary crossentropy loss. So, a network parameter trend in multiclass classification is also seen in the IoT malware detection case.

5.2. t-SNE features visualization

The DL model's internal learning process of the input data and achieved DL model performance is difficult to explain due to high dimensional data processed by these models. t-distributed stochastic neighbor embedding (t-SNE) statistical method may be used to reduce the high dimensionality data and can visualize the reduced dimensions. t-SNE uses principal component analysis (PCA) for reducing higher dimensions to two-dimensional data. To understand how the GRU models synthesize the data and classify the IoT malware and benignware samples, we have utilized t-SNE to represent high dimensional data into two dimensions. The penultimate layer of the models is passed through the t-SNE to obtain the dimension reduction output.

The Figs. 8, 9 and 10 represents the t-SNE feature visualization of the GRU, Bi-GRU, and Bi-GRU-CNN models for IoT malware binary classification or IoT malware detection. As illustrated in the Fig. 8, the GRU model clearly distinguishes the malware and benignware represented in red and green color clusters are separated by a noticeable distance except few overlapped small clusters in the middle region of the plot. These feature cluster separation represents that the GRU model can learn the structure and semantics of the malware byte sequences. So, we can convince that the GRU model learns the malware byte sequences for binary classification.

The t-SNE representation of the Bi-GRU shows that there are more overlapped small regions compared to the GRU model. The separation between the malware and benignware is difficult in such regions and hence the Bi-GRU performance slightly diminished compared to the GRU model. However, some regions highlighted in red color representing malware clearly separated from the benignware region and results in good classification of samples.

The Bi-GRU-CNN model t-SNE representation in the Fig. 10 shows that small size malware and benignware clusters formed compared to the GRU model. But, we could still see that the possibility of separation between malware and benignware within the cluster regions. The spatial temporal feature extraction of the CNN along with the time sequence data extraction makes small clusters formed in the model. However, we could still see that some overlap of the malware and benignware spread all over the plot. This results in the model still not achieved 100% accuracy for malware binary classification.

The IoT malware family classification feature visualization for the GRU, Bi-GRU, and Bi-GRU-CNN models are shown in Figs. 11, 12 and 13. There is a clear separation between benignware and malware class samples represented in the low dimension form for all the three models such as GRU, Bi-GRU, and Bi-GRU-CNN. This indicates that GRU based models can achieve good prediction performance. From the Figs. 11, 12 and 13, we can also see that the

Table 11
Hyperparameters used in the DL models for Malware multiclass classification performance evaluation .

Parameters	RNN	LSTM	GRU	Bi-RNN	Bi-LSTM	Bi-GRU	Bi-GRU-CNN
Total param	50,569	149,257	116,745	91,657	289,033	224,009	273,289
Trainable params	50,569	149,257	116,745	91,657	289,033	224,009	273,289
Non-trainable params	0	0	0	0	0	0	0
Dense	64,9	64,9	64,9	64,9	64,9	64,9	64,9
Dropout	0.1, 0.1,0.1	0.1, 0.1,0.1	0.1, 0.1,0.1	0.1, 0.1,0.1	0.1, 0.1,0.1	0.1, 0.1,0.1	0.1, 0.1,0.1, 0.1
Activation	ReLU, Softmax	ReLU, Softmax	ReLU, Softmax	ReLU, Softmax	ReLU, Softmax	ReLU, Softmax	ReLu, ReLU, Softmax
Batchsize	64	64	64	64	64	64	64
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam
Loss	Categorical	Categorical	Categorical	Categorical	Categorical	Categorical	Categorical
Number of Units	128	128	128	128	128	128	128
Epochs	40	40	40	40	40	40	40
Flatten	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Conv1D	-	-	-	-	-	-	Yes
MaxPooling1D	-	-	-	-	-	-	Yes
Filters	-	-	-	-	-	-	128

bashlite, mirai, and tsunami malware families are overlapped in few cluster area regions of all three models t-SNE visualization plots. This is an indication that similar functionality IoT malware families classification is tricky and requires the model need to understand not only the byte sequences but also additional features as well. The android malware family features are overlapped with benignware in the GRU case, as shown in the Fig. 11 left corner. The bidirectional GRU has separation issues and we can clearly see in Fig. 12, the malware families still overlap in the clustering region. Based on the Fig. 12, the fine-grained separation between the malware families needed to achieve the best performance accuracy. On the contrary, as shown in Fig. 13 the Bi-GRU-CNN model separates the mirai, tsunami, and bashlite to a great extent compared to the other two models. Hence, the performance results obtained from Bi-GNN-CNN are better than the other two models.

5.3. Meantime to detection(MTTD)

In general, the MTTD in incident management is measured as the average time elapsed between the incident's start time and the incidents detection time. The malware MTTD for machine learning or deep learning based detection methods is defined as the average time elapsed between a malware sample is ready to feed into the trained model and the model predicts the malware sample is indeed malware. To evaluate the effectiveness of the proposed model, we measured the malware MTTD on the Bi-GRU-CNN model binary classification scenario. A randomly selected few hundred malware test samples were used to measure the time taken to complete the malware sample accurate prediction. This process has been been repeated more than 100 times to measure the average detection time. Our results indicate that the malware MTTD using deep learning based detection is 0.245 Sec. A few hundreds of milliseconds range MTTD value clearly indicate that our trained model can be used for real time IoT malware sample detection.

5.4. Discussion and limitations

The static feature based byte sequences from the program entry point in the ELF binary file as a feature is applied to the proposed deep learning approach Bi-GRU-CNN achieved promising results for both IoT malware detection and family classification. Our approach don't require preprocessing the file to collect dynamic features like API call or static features such as Opcodes or CFG features, which are mostly used in the prior art to apply ML/DL models for solving the IoT malware detection and classification problems. Our feature selection process requires minimal changes to collect the byte sequences from the binary files. The performance evaluation showed that the proposed DL model Bi-GRU-CNN achieved accuracy 100% for malware binary classification and 98% for malware family classification when we use byte sequences and CPU type as a feature. We also don't require feature engineering, as we use the raw byte sequences from the binary sample file. Interestingly, when we use only byte sequence as input, our model still able to achieve good performance. It signifies that our approach is robust, platform independent and has the capability to accurately classify the IoT malware families supported on heterogeneous CPU architectures.

We believe that the byte sequence from the entry point approach can be more suitable for IoT malware detection and classification, as the majority of the IoT malware are variants of the well known malware family Bashlite, Mirai, and the adversary only add some minor functionality or change few areas in the source code. The chance of modifying the code around the main function is unlikely if we consider the malware is written in "C" programming code. But, a well known and sophisticated attacker who is aware of using byte sequence as a feature set can modify the source code and can evade the detection. So, an attacker may perform adversarial attack to evade the detection. One of the future work will be evaluating our model in adversarial attack scenarios where an adversary constantly pursing byte sequence based model detection evading.

Our dataset also include an Android based malware class to validate the heterogeneous malware samples representing not only elf executables written in C but also APK files written in Java programming language. As the IoT applications can also be developed and run as an Android application, the Android malware class also being considered as one of the malware class. Since the proportion of the Android malware samples are much lower than the IoT malware samples in the dataset and the CPU families are also being considered as an input parameter, the potential bias introduced by the Android malware samples is very low. Although the Android applications developed in Java, we believe that the Android malware class inclusion covers another set of IoT application files used in Android supported operating systems.

Our model may not be able to classify the malware families, which are small proportions in the overall dataset because of imbalance in datasets. So, some malware like xorddos, pnscan were difficult to classify using the proposed model. Proposing new models for handling imbalance datasets in IoT space and building the balanced multiclass IoT datasets is another future direction of our work. We are also working on including cost-sensitive learning i.e. giving higher weights to the classes that contain less number of data samples and lower weights to the class that contains higher data samples. Cost-weight selection is done by using genetic algorithm.

The proposed model can be implemented as a standalone cloud solution to detect and classify the IoT malware. MLops stack can be

used to implement our model in cloud environment and used as an open-source service to identify the IoT malware class, given the unknown file as input. In order to detect the malware near the IoT device environment, the federated learning based approach can be followed to consider the samples in each IoT node or IoT gateway and classify the IoT malware. This type of work is considered as one of the future work to extend the current work.

The dynamic analysis based feature extraction and leveraging hybrid analysis features for ML/DL based IoT malware detection and classification are still in infancy stage. This may be due to the complexity of setting up the virtual environment supporting heterogeneous CPU architectures. However, we may use the QEMU processor emulator and OpenWRT open source operating system to run IoT malware in multiple CPU architectures (Pa Pa et al., 2015) for dynamic analysis. One of research direction will be proposing new approaches in this area to accurately detect and classify the IoT malware while not penalizing the detection methods with training and prediction time delay.

6. Conclusion

IoT malware detection has been a concern, as these devices operate with low power, low memory, and computation capability and makes it difficult to deploy the security monitoring tools or monitoring services in the device network premises. In this article, we have proposed effective, robust, and platform independent DL model Bi-GRU-CNN based approach to perform the IoT malware classification. The static analysis byte sequences from the entry point of the ELF binary file sample is used as an input feature for applying to DL model. Additionally, we have showcased the detail performance analysis of the various DL models for IoT malware detection and malware family classification capabilities when byte sequences are used as an input feature. Our performance investigation showed promising results for IoT malware detection with 100% detection accuracy and malware family classification with 98% detection accuracy when the proposed approach Bi-GRU-CNN is applied. But, some malware families were not be able to correctly classified in the dataset, as the datasets are imbalanced. So, further study is required on the proposed model to interpret the imbalance dataset classification capabilities and proposing new solutions for handling imbalance datasets. We are also interested in studying the impact of IoT adversarial attacks on the Bi-GRU-CNN model performance.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Rajasekhar Chaganti: Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing, Validation. **Vinayakumar Ravi:** Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing, Validation. **Tuan D. Pham:** Writing – review & editing, Supervision.

References

Alasmary, H., Khormali, A., Anwar, A., Park, J., Choi, J., Abusnaina, A., Awad, A., Nyang, D., Mohaisen, A., 2019. Analyzing and detecting emerging internet of things malware: a graph-based approach. IEEE Internet Things J. 6 (5), 8977–8988. doi:10.1109/JIOT.2019.2925929.

Alhanahnah, M., Lin, Q., Yan, Q., Zhang, N., Chen, Z., 2018. Efficient signature generation for classifying cross-architecture IoT malware. In: 2018 IEEE Conference on Communications and Network Security, CNS 2018, pp. 1–9. doi:10.1109/CNS.2018.8433203.

Alrawi, O., Lever, C., Valakuzhy, K., Court, R., Snow, K., Monrose, F., Antonakakis, M., 2021. The circle of life : a large-scale study of the IoT malware lifecycle. Usenix'21.

Baek, S., Jeon, J., Jeong, B., Jeong, Y.-s., 2021. Two-stage hybrid malware detection using deep learning. Hum. Centric Comput. Inf. Sci..

Bakhshinejad, N., Hamzeh, A., 2019. Resilient and Deep Network for Internet of Things (IoT) Malware Detection, Vol. 1150 CCIS. Springer Singapore doi:10.1007/978-981-15-1960-4_13.

Bazrafshan, Z., Hashemi, H., Fard, S.M.H., Hamzeh, A., 2013. A survey on heuristic malware detection techniques. In: The 5th Conference on Information and Knowledge Technology. IEEE, pp. 113–120.

Boppana, R.V., Chaganti, R., Vedula, V., 2020. Analyzing the Vulnerabilities Introduced by DDoS Mitigation Techniques for Software-Defined Networks, Vol. 1055. Springer International Publishing doi:10.1007/978-3-030-31239-8_14.

Charlie Osborne, F., 2017. Researchers discover over 170 million exposed IoT devices in major US cities | ZDNet. https://www.zdnet.com/article/researchers-expose-vulnerable-iot-devices-in-major-us-cities/.

Carrillo-Mondéjar, J., Martínez, J.L., Suarez-Tangil, G., 2020. Characterizing Linux-based malware: findings and recent trends. Future Gener. Comput. Syst. 110, 267–281. doi:10.1016/j.future.2020.04.031.

Center, T. I. S., 2021. TWISC research centers. https://www.twisc.org/research-centers/.

Darabian, H., Dehghantanha, A., Hashemi, S., Homayoun, S., Choo, K.K.R., 2020. An opcode-based technique for polymorphic internet of things malware detection. Concurrency Comput. 32 (6). doi:10.1002/cpe.5173.

Dib, M., Torabi, S., Bou-Harb, E., Assi, C., 2021. A multi-dimensional deep learning framework for IoT malware classification and family attribution. IEEE Trans. Netw. Serv. Manage. 18 (2), 1165–1177. doi:10.1109/TNSM.2021.3075315.

Dima Ben, I., 2016. Breaking down mirai: an IoT DDoS botnet analysis. https://www.imperva.com/blog/malware-analysis-mirai-ddos-botnet/.

Dovom, E.M., Azmoodeh, A., Dehghantanha, A., Newton, D.E., Parizi, R.M., Karimipour, H., 2019. Fuzzy pattern tree for edge malware detection and categorization in IoT. J. Syst. Archit. 97, 1–7. doi:10.1016/j.sysarc.2019.01.017.

D'Angelo, G., Ficco, M., Palmieri, F., 2020. Malware detection in mobile environments based on autoencoders and API-images. J. Parallel Distrib. Comput. 137, 26–33.

Fredrik Dahlqvist, 2019. Growing opportunities in the internet of things | McKinsey. https://www.mckinsey.com/industries/private-equity-and-principal-investors/our-insights/growing-opportunities-in-the-internet-of-things#.

Gaur, P., Tahiliani, M.P., 2015. Operating systems for IoT devices: a critical survey. In: Proceedings - 2015 IEEE Region 10 Symposium, TENSYMP 2015, pp. 33–36. doi:10.1109/TENSYMP.2015.17.

HaddadPajouh, H., Dehghantanha, A., Khayami, R., Choo, K.K.R., 2018. A deep recurrent neural network based approach for internet of things malware threat hunting. Future Gener. Comput. Syst. 85, 88–96. doi:10.1016/j.future.2018.03.007.

Hwang, C., Hwang, J., Kwak, J., Lee, T., 2020. Platform-independent malware analysis applicable to windows and Linux environments. Electronics 9 (5). doi:10.3390/electronics9050793.

Jeon, J., Park, J.H., Jeong, Y.S., 2020. Dynamic analysis for IoT malware detection with convolution neural network model. IEEE Access 8, 96899–96911. doi:10.1109/ACCESS.2020.2995887.

Jgamblin, 2017. Mirai-Source-Code: leaked mirai source code for research/IoC development purposes. https://github.com/jgamblin/Mirai-Source-Code.

Kumar, A., Lim, T.J., 2019. EDIMA: early detection of IoT malware network activity using machine learning techniques. In: IEEE 5th World Forum on Internet of Things, WF-IoT 2019 - Conference Proceedings. Institute of Electrical and Electronics Engineers Inc., pp. 289–294. doi:10.1109/WF-IoT.2019.8767194.

Lecun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444. doi:10.1038/nature14539.

Linuxfoundation, 2021. ELF header. https://refspecs.linuxfoundation.org/elf/gabi4+/ch4.eheader.html.

Loeb, L., 2016. BASHLITE malware uses IoT for DDoS attacks. https://securityintelligence.com/news/bashlite-malware-uses-iot-for-ddos-attacks/.

Namavar Jahromi, A., Hashemi, S., Dehghantanha, A., Choo, K.K.R., Karimipour, H., Newton, D.E., Parizi, R.M., 2020. An improved two-hidden-layer extreme learning machine for malware hunting. Comput. Secur. 89. doi:10.1016/j.cose.2019.101655.

Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.S., 2011. Malware images: visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security, pp. 1–7.

Nghi Phu, T., Dai Tho, N., Huy Hoang, L., Ngoc Toan, N., Ngoc Binh, N., 2021. An efficient algorithm to extract control flow-based features for IoT malware detection. Comput. J. 64 (4), 599–609. doi:10.1093/comjnl/bxaa087.

Nguyen, H.-T., Ngo, Q.-D., Le, V.-H., 2018. IoT botnet detection approach based on psi graph and DGCNN classifier. In: 2018 IEEE International Conference on Information Communication and Signal Processing (ICICSP). IEEE, pp. 118–122.

Nguyen, H.T., Ngo, Q.D., Le, V.H., 2020. A novel graph-based approach for IoT botnet detection. Int. J. Inf. Secur. 19 (5), 567–577. doi:10.1007/s10207-019-00475-6.

Niu, W., Zhang, X., Du, X., Hu, T., Xie, X., Guizani, N., 2019. Detecting malware on X86-based IoT devices in autonomous driving. IEEE Wirel. Commun. 26 (4), 80–87. doi:10.1109/MWC.2019.1800505.

Pa Pa, Y.M., Suzuki, S., Yoshioka, K., Matsumoto, T., Kasama, T., Rossow, C., 2015. IoTPOT: analysing the rise of IoT compromises. 9th USENIX Workshop on Offensive Technologies, WOOT 2015.

PELAEZ, A., 2021. 9 IoT operating systems to use in 2021. https://ubidots.com/blog/iot-operating-systems/.

Phu, T.N., Dang, K.H., Quoc, D.N., Dai, N.T., Binh, N.N., 2019. A novel framework to

classify malware in MIPS architecture-based IoT devices. Secur. Commun. Netw. 2019. doi:10.1155/2019/4073940.
Raff, E., Barker, J., Sylvester, J., Brandon, R., Catanzaro, B., Nicholas, C.K., 2018. Malware detection by eating a whole EXE. In: Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence.
Raju, A.D., Abualhaol, I.Y., Giagone, R.S., Zhou, Y., Huang, S., 2021. A survey on cross-architectural IoT malware threat hunting. IEEE Access 9, 91686–91709. doi:10.1109/access.2021.3091427.
Rathore, H., Agarwal, S., Sahay, S.K., Sewak, M., 2018. Malware detection using machine learning and deep learning. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11297 LNCS, pp. 402–411. doi:10.1007/978-3-030-04780-1_28.
Samantray, O.P., Tripathy, S.N., 2021. IoT-Malware Classification Model Using Byte Sequences and Supervised Learning Techniques. Springer Singapore doi:10.1007/978-981-16-0666-3_6.
Scott, J., 2017. Signature Based Malware Detection is Dead. Institute for Critical Infrastructure Technology.
Krebs on Security, 2016. Source code for IoT botnet 'Mirai' released. https://krebsonsecurity.com/2016/10/source-code-for-iot-botnet-mirai-released/.
Shahzad, F., Farooq, M., 2012. ELF-Miner: using structural knowledge and data mining methods to detect new (Linux) malicious executables. Knowl. Inf. Syst. 30 (3), 589–612. doi:10.1007/s10115-011-0393-5.
Shalaginov, A., Øverlier, L., 2021. A novel study on multinomial classification of x86/x64 Linux ELF malware types and families through deep neural networks. In: Malware Analysis Using Artificial Intelligence and Deep Learning. Springer International Publishing, pp. 437–453. doi:10.1007/978-3-030-62582-5_17.
Venkatraman, S., Alazab, M., Vinayakumar, R., 2019. A hybrid deep learning image-based analysis for effective malware detection. J. Inf. Secur. Appl. 47 (March 2020), 377–389. doi:10.1016/j.jisa.2019.06.006.
Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A., Venkatraman, S., 2019. Deep learning approach for intelligent intrusion detection system. IEEE Access 7, 41525–41550. doi:10.1109/ACCESS.2019.2895334.
Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Venkatraman, S., 2019. Robust intelligent malware detection using deep learning. IEEE Access 7 (March 2020), 46717–46738. doi:10.1109/ACCESS.2019.2906934.
Vinayakumar, R., Soman, K.P., 2018. DeepMalNet: evaluating shallow and deep networks for static PE malware detection. ICT Express 4 (4), 255–258. doi:10.1016/j.icte.2018.10.006.
Wan, T.-L., Ban, T., Cheng, S.-M., Lee, Y.-T., Sun, B., Isawa, R., Takahashi, T., Inoue, D., 2020. Efficient detection and classification of internet-of-things malware based on byte sequences from executable files. IEEE Open J. Comput. Soc. 1 (November), 262–275. doi:10.1109/ojcs.2020.3033974.
Wan, T.L., Ban, T., Lee, Y.T., Cheng, S.M., Isawa, R., Takahashi, T., Inoue, D., 2020. IoT-Malware detection based on byte sequences of executable files. In: Proceedings - 2020 15th Asia Joint Conference on Information Security, AsiaJCIS 2020, pp. 143–150. doi:10.1109/AsiaJCIS50894.2020.00033.

Rajasekhar Chaganti work as a security engineer in security team at Expediagroup Inc to detect, mitigate and respond to the security threats in large scale enterprise organization. Currently, he is also involved in security research and working towards PhD degree from University of Texas San Antonio. Prior to joining Expedia, He served as a security analyst in procore technologies and possess the knowledge on SIEM, vulnerability management, DNS and endpoint security. He is holding the industry certifications such as AWS Solution Architect, Certified Ethical Hacker, and CCNA cyberops covering various aspects of security. He was a research assistant at UTSA performing software defined networking security research and received MS in computer science focusing on computer and Information security from UTSA in 2018. His research Interests are in machine learning/AI for security applications, network security, threat detection in IoT, cloud and blockchain environments and social engineering scams. Prior to joining UTSA, he worked as a patent research analyst for stellarix consulting services in Jaipur, India. He performed patent research related to patent novelty, invalidation and landscape projects in electronics and computer science domain focusing on the cybersecurity, machine learning, IOT security and cloud computing. He has been a reviewer for IEEE Access, Information and computer security, springer cybersecurity Journal, wireless communications and mobile computing, journal of cybersecurity and mobility, cybersecurity skills journal and volunteered for organizing cybersecurity symposium for smart cities and other security initiatives.

Vinayakumar Ravi is an Assistant Research Professor at Center for Artificial Intelligence, Prince Mohammad Bin Fahd University, Khobar, Saudi Arabia. My previous position was a Postdoctoral research fellow in developing and implementing novel computational and machine learning algorithms and applications for big data integration and data mining with Cincinnati Children's Hospital Medical Center, Cincinnati, OH, USA from September, 2019 to September, 2020. He received the Ph.D. degree in computer science from Computational Engineering & Networking, Amrita School of Engineering, Coimbatore, Amrita Vishwa Vidyapeetham, India. His Ph.D. work centers on Application of Machine learning (sometimes Deep learning) for Cyber Security and discusses the importance of Natural language processing, Image processing and Big data analytics for Cyber Security. His current research interests include applications of data mining, Artificial Intelligence, machine learning (including deep learning) for biomedical informatics, Cyber Security, image processing, and natural language processing. More details available at https://vinayakumarr.github.io/. He has more than 50 research publications in reputed IEEE conferences, IEEE Transactions and Journals. His publications include prestigious conferences in the area of Cyber Security, like IEEE S&P and IEEE Infocom. He has given many invited talks on deep learning applications in IEEE conferences and Industry workshops in 2018. He has got a full scholarship to attend Machine Learning Summer School (MLSS) 2019, London. Dr. Ravi has served as a Technical Program Committee (TPC) member at international conferences including SSCC Symposium, IEEE TrustCom-2020, and IEEE SmartData-2020. He is an editorial board member for Journal of the Institute of Electronics and Computer (JIEC), International Journal of Digital Crime and Forensics (IJDCF), and he has organized a shared task on detecting malicious domain names (DMD 2018) as part of SSCC'18 and ICACCI'18. He received the Chancellor's Research Excellence Award in AIRA 2021 and was included in the World's Top 2 Scientists by Stanford University published in PLoS Biology.

Tuan D. Pham currently holds positions as (full) Senior Research Professor in AI and Founding Director of the Center for Artificial Intelligence at Prince Mohammad Bin Fahd University, Saudi Arabia. His previous position was (full) Professor of Biomedical Engineering at Linkoping University, University Hospital Campus, Linkoping, Sweden. He was appointed as (full) Professor and Leader of the Aizu Research Cluster for Medical Engineering and Informatics, and the Medical Image Processing Lab, both at the University of Aizu, Japan. Before his appointments in Japan, he was appointed as Associate Professor and the Bioinformatics Research Group Leader at the University of New South Wales, Canberra, Australia. His current research focuses on AI and machine learning methods for image processing, time-series analysis, complex networks, and pattern recognition applied to medicine, biology, and mental health. He serves as an Associate/Section Editor for a number of scholarly journals, series, and conference proceedings, such as Pattern Recognition (Elsevier), Heliyon (Cell Press), IET Signal Processing, Entropy (MDPI), Frontiers in Artificial Intelligence, Frontiers in Big Data, Computer Science Advisory Board (Cambridge Scholars), Current Bioinformatics (Bentham), IEEE-EMBC (Theme 10: Biomedical & Health Informatics), ACM, and SPIE conference proceedings. Dr. Pham has been selected as an Expert in Artificial Intelligence for consultation by the U.S. Food & Drug Administration (FDA) Center for Devices and Radiological Health (CDRH) Network of Digital Health Experts Program (NoDEx), and included in the World's Top 2% Scientists by Stanford University published in PLoS Biology