Received June 23, 2020, accepted July 9, 2020, date of publication July 16, 2020, date of current version July 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3009819

A Dynamic DL-Driven Architecture to Combat Sophisticated Android Malware

IRAM BIBI[1], ADNAN AKHUNZADA[2], JAHANZAIB MALIK[3], JAVED IQBAL[1], ARSLAN MUSSADDIQ[4], AND SUNGWON KIM[4]

[1]Computer Science Department, COMSATS University, Islamabad 46000, Pakistan
[2]Technical University of Denmark, 2800 Copenhagen, Denmark
[3]National Cyber Security Auditing and Evaluation Laboratory, NUST, Islamabad 44000, Pakistan
[4]Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, South Korea

Corresponding author: Sungwon Kim (swon@yu.ac.kr)

This research was supported in part by the Brain Korea 21 Plus Program (22A20130012814) funded by the National Research Foundation of Korea (NRF), in part by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2020-2016-0-00313) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation), and in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2018R1D1A1A09082266).

ABSTRACT The predominant Android operating system has captured enormous attention globally not only in smart phone industry but also for varied smart devices. The open architecture and application programming interfaces (APIs) while hosting third party applications has led to explosive growth of varied pervasive sophisticated Android malware production. In this study, we propose a robust, scalable and efficient Cuda-empowered multi-class malware detection technique leveraging Gated Recurrent Unit (GRU) to identify sophisticated Android malware. Experimentation of the proposed technique has been carried out using current state-of-the-art datasets of Android applications (i.e., Android Malware Dataset (AMD), Androzoo). Moreover, to rigorously evaluate the performance of the proposed technique, we have employed standard performance evaluation metrics (e.g., accuracy, precision, recall, F1-score etc.) and compared it with our constructed DL-driven architectures and benchmark algorithms. The GRU-based malware detection system outperforms with 98.99% detection accuracy for malware identification with a trivial trade off in speed efficiency.

INDEX TERMS Android malware, deep learning, recurrent neural network, convolutional neural network, deep neural network, mobile security.

I. INTRODUCTION

The evolution from traditional to smart devices has revolutionized the world. Currently, an exponential growth of smart devices has been witnessed both for personal and commercial use. According to a report published by Gartner [1], approximately 400 million smartphones were sold in 2019. By analysing smartphones popularity, stakeholders have also shown strong interest towards developing proprietary mobile operating systems (OS) [2]. Android being an open source and versatile platform is considered as a leading giant in the telecommunication industry and a de facto standard for various smart phone manufacturers. In 2019, Android is the most predominant OS; holding an approximately 74% of market shares around the globe in smart phone industry [3]. Additionally, apart from smart phones Android is also taking over smart watches, tablets, televisions, digital boxes and so forth.

The associate editor coordinating the review of this manuscript and approving it for publication was Shadi Aljawarneh.

Due to the prevailing environment of Android, it is becoming a potential target for cyber adversaries. Malware developers are essentially motivated of crafting sophisticated malwares to exploit existing OS vulnerabilities [4], [5]. The term malware is a collection of malicious software variants intentionally designed (i.e., Viruses, Trojans, Spyware, Adware and Ransomware etc.) [6] to cause extensive damage to data and system as privilege escalation, information theft, remote control and privacy breach etc [7]. Sophisticated malware is highly impeccable and can simply throw the whole industry into chaos. Moreover, malware growth continues to increase by the fact that Android malware can immensely influence both enterprise and end-users. The literature is evident that malware can possibly get Android root access and is not traceable subsequently [8].

Cyber attackers manage to produce unprecedented levels of disruption such as exploitation of zero-day vulnerabilities leveraging diverse tools and tactics [4] to disrupt various systems. To reverse the effects of ever evolving cyber threats, the situation makes malware detection techniques worth studying and improving. Deep learning techniques can help tackle dynamic evolving malware in Android environment timely and efficiently [9], [10]. In this paper, we propose an efficient and scalable Deep Learning (DL) based malware detection scheme employing Gated Recurrent Unit (GRU) to comprehensively detect varied Android malware timely and efficiently.

A. CONTRIBUTIONS

The main contributions of the proposed study are manifold:

- The authors propose a flexible, innovative and scalable DL-based detection mechanism leveraging Recurrent Neural Network (RNN), particularly Gated Recurrent Unit (GRU) to identify multi-class attacks effectively in Android environment.
- To comprehensively evaluate multi-class attacks and evolving malwares in Android, a current state-of-the-art publicly available android datasets (i.e., AMD, Androzoo) have been employed.
- Standard performance evaluation metrics (i.e., accuracy, precision, recall, F1-score etc.) have been used to thoroughly evaluate our proposed mechanism. Moreover, our proposed technique outperforms in terms of detection accuracy with a trivial trade-off in computational complexity.
- Furthermore, we also compare our proposed technique with our constructed other DL-driven algorithms (i.e., Long short-term memory, Convolutional Neural Network, and Deep Neural Network) and current benchmarks.

B. ORGANIZATION

The remaining paper is adjusted in the following way; section II shows the relevant literature. Section III is about the detailed architectural description of utilised algorithms. Section IV includes of a detailed overview of our proposed system (i.e., system design, dataset, and constructed algorithms). Section V discusses experimental results and our findings from evaluation. Finally, Section VI concludes the paper and defines future directions.

II. RELATED WORK

In recent work, researchers have proposed different malware detection framework to protect against ever evolving sophisticated malware in Android. The majority of the current works are Artificial Intelligence-based binary malware recognition systems. However, deep learning (DL) structures like Convolutional Neural Network (CNN), Deep Belief Network (DBN), Recurrent Neural Networks (RNN), Deep Neural Network (DNN) are still in its earliest stages towards comprehensive evaluation of lethal multi-class attacks for Android platform.

In [11], *millar et al.* proposed Android malware detection framework leveraging Discriminative Adversarial Network (DAN) for classification of obfuscated and un-obfuscated applications as malicious or benign. Practical experimentation is performed on Drebin dataset and achieved 97% average detection accuracy. *Lee et al.* in [12], presented a malware identification scheme employing Gated Recurrent Unit (GRU) and Convolutional Neural Network (CNN). The experiment has been performed on about two million samples collected from VirusTotal and achieved detection accuracy of 97.7%. The study [13] implemented Deep Neural Network (DNN) architecture to classify malicious Android applications. An Android package kit (APK) based dataset for binary classification have been employed for evaluation for the proposed framework that achieves 95% detection accuracy. *Alzaylaee et al.* in [14], proposed a DL-Droid framework to detect malicious Android applications through using a input generation scheme for efficient code coverage and to improve performance. Features are extracted from Android applications and genuine API calls. The experiment is performed on total 31,125 Apk's, where 11,505 were malwares and 19,620 benign samples were taken from Intel-Security (McAfee Labs). The proposed framework acquires 95.2% detection accuracy. Additionally, in [15], the authors presented a malware detection framework using system calls while employing Long short-term memory (LSTM). The employed dataset (i.e., Drebin) contain 3567 Malicious and 3536 Benign applications and obtain detection accuracy of 93.7%.

Moreover, real time automated framework has been developed in [16] to classify malicious and benign Android applications. The proposed Gated Recurrent Unit (GRU) based framework achieved detection accuracy of 91.42% utilizing Contagio malware dump dataset.

C. Hasegawa et al. proposed a lightweight malware identification technique leveraging Convolutional Neural Network (CNN) to analyse raw APK's in [17]. The dataset considered in this study contains 5000 malware and 2000 benign Android applications from AMD and Drebin dataset. The framework secures 97% of an average detection accuracy while employing a 10-fold cross validation technique. Using permissions and API call's, [18] proposed Deep Neural Network (DNN) to classify Android malicious applications. The framework outperforms with 97% detection accuracy on Drebin dataset. To detect Android sophisticated Malware, *Karbab et al.* presented a system called MalDozer [19] to determine malware sequence from raw data of different API calls. The proposed system employs Artificial Neural Network (ANN) and acquire 90% of detection accuracy. The utilized datasets for proposed framework are Malgenome-2015 with 1K samples, Drebin-2015 with 5.5K samples, MalDozer dataset with 20K samples. Moreover, 38k benign Android applications are downloaded from google play store. To convert the APK's binary data into images, the study presented the image texture-based classification mechanism in [20]. The technique utilizes Deep Belief Network (DBN) and extracts features like API calls, permissions and activities. The framework is experimented on Drebin dataset and the achieved results shows that image texture when combined with API calls gets 95.6% detection accuracy. In [21], *Zhang et al.* proposed DeepclassifyDroid that is based on Convolutional Neural Network (CNN) for Android malware detection. The dataset containing 5546 malicious and 5224 benign applications achieves 97.4% accuracy which also reveals lack in system performance and need of dataset with enough instances present. However, [22] establish a DDefender technique for malware identification using Deep Neural Network (DNN) for dynamic as well as static analysis. The dataset comprised of 4208 Android applications instances and achieved 95% detection accuracy which is not good enough for real time deployment of system. *Li et al.* in [23] implemented a malware identification system based on API calls and permission to design a Deep Belief Network (DBN). The proposed scheme considered Drebin dataset for experimentation and achieves 90% detection accuracy.

Moreover, Long short-term memory (LSTM) in [24] is employed for malware characterization in Android. The dataset containing 1738 records of Android applications that obtain 93.9% and 97.5% detection accuracy for dynamic and static analysis separately. The study [25] suggests Deep-Refiner malware identification framework likewise automates feature extraction process leveraging Long short-term memory (LSTM). Deep-Refiner considers dataset with 62,915 malware and 47,525 benign Android Apk's. The achieved detection accuracy of proposed technique is 97.74% that is still not adequate for real time deployment of the proposed framework. Consequently, Convolutional Neural Network (CNN) based Android malware detection system represents Android applications into RGB colour code is presented in [26]. The encoded image is passed as an input to CNN classifier for automation of feature extraction and execution process. The proposed approach achieved 97.25% detection accuracy for 829356 android application of leopard mobile Inc. The presented technique in [27] utilized diverse DL-based classifiers (i.e., CNN, RNN, DAE, DBN and LSTM) to efficiently detect Android malware by considering the requested permissions, components and filtered intents. For practical experimentation Drebin and Virus-share dataset achieves 93.6% detection accuracy for LSTM. To detect sophisticated malwares, [28] proposed multimodal Convolutional Neural Network (CNN) to reflect the properties of Android applications as malevolent or benign. Feature vector is extracted through similarity-based feature extraction method while using Permissions, API calls and so forth. The dataset comprises of 41260 applications (i.e., Virus-Share, Malgenome Project, Google Play App Store and Virus-Total) and achieved 98% detection accuracy.

Reference [29] presented the technique based on Natural Language processing (NLP) for Android malware analysis. The proposed system considers system call sequence as text file. The dataset contains 14231 applications in total from Android virtual device (AVD) system calls dataset and acquire an accuracy of 93.16%. *Singh et al.* presented the conventional machine learning and deep learning models in [30] for pervasive malware detection. Considered algorithms are Decision tree, Random forest, Gradient boosted trees, K-NN, Deep Neural Network (DNN) and SVM. However, SVM performs better having 97.16% of detection accuracy. The utilised dataset comprises of small number of instances (i.e., 494 applications). Additionally, [31] develop Auto-Droid (automatic Android malware detection) rather than depending on API call mentioned in manifest file, the technique used API calls of smali code at the kernel of Android OS. The model gets performance of 95.98% using Deep Belief Network (DBN) and Stacked Auto-Encoders (SAE) for newly unknown malware detection.

Reference [32] presented Deep Flow based Deep Belief Network (DBN) for malware identification based on data streams for malignant applications that differ from benign applications. The proposed system is executed for 3000 benign and 8000 malicious applications and obtains an accuracy of 95.05%. To explore the ability of Convolutional Neural Network (CNN) for malware detection through conversion of bytecode into text in [33]. The proposed framework has an edge to extract features automatically from raw data. The experiment performs on diverse datasets: Genome project and McAfee Labs dataset. The model achieves accuracy of 98% by utilizing Genome dataset and 87% accuracy for McAfee dataset. Adversary resistant technique developed in [34] that obstructs adversary from constructing adversarial samples. The DNN-based framework is experimented employing a dataset of 14,679 malware and 17,399 benign variants from MNIST CIFAR-10. The presented technique attain accuracy of 95.2% detection accuracy. The framework in [35] converts Android permissions into features leveraging LSTM layer and bag-of-words technique. Cyber Security Data Mining Competition (CDMC2016) dataset has been utilized and achieved 89.7% detection accuracy.

The given literature comparison (see Table 1) is quite evident that DL models are still in its commencement towards comprehensive evaluation of diverse multi-attacks in Android environment. Clearly, most of the existing mechanisms are devised for binary classification only. Further, this research came up with the idea of cuda-empowered GRU-based detection framework to extensively evaluate DL-models for the detection of sophisticated Android malware.

III. PRELIMINARIES

The basic architecture and explanation of the utilized varied DL-algorithms (i.e., Gated Recurrent Unit, Long short-term memory (LSTM), Convolutional Neural Network (CNN) and Deep Neural Network (DNN)) have been briefly explained.

A. GATED RECURRENT UNIT (GRU)

Gated Recurrent Unit (GRU), is a powerful variant of standard Recurrent Neural Network (RNN) and similar to an LSTM utilise the combined gating mechanism as a solution to short-term memory. The GRU has an internal mechanism

TABLE 1. Selected previous studies for android malware detection.

REF	YEAR	FEATURES	DATASET	Model	CONTRIBUTION	LIMITATION
[11]	2020	Opcodes, permissions, APIcalls	Drebin-68,880 applications	DAN	Android malware detection framework leveraging Discriminative Adversarial Network(DAN) that classifies obfuscated and un-obfuscated apps and achieved 97% average detection accuracy.	Not performance efficient for real time deployment.
[12]	2019	Permissions, Intents	APK files	GRU, CNN	Proposed a DL-based framework for identification of Android malware employing GRU and CNN.	Not satisfactory performance for real time deployment.
[13]	2019	Permission extracted by using Android Asset Packaging Tool.	Android APK files	DNN	Implemented a DL-based technique for the classification of Android malware and achieves 95% detection accuracy.	The detection accuracy is not good enough for malware detection to provide secure mechanism.
[14]	2019	Features(dynamic) extracted from applications through smartphones i.e., permission, API calls	31,125 Apks; 11,505 malware and 19,620 benign samples obtained from Intel Security (McAfee-Labs).	DNN	Proposed DL-Droid system to detect Android malicious applications through analysis using stateful input generation and achieves 95.2% accuracy.	Need to investigate other DL-based classifiers to improve detection accuracy.
[15]	2019	system call sequence	3567 Malicious and 3536 Benign applications from Drebin	LSTM	framework uses system call as a sentence in language and proposed a LSTM-based deep learning framework.	Need to investigate other DL-based classifiers to improve detection accuracy.
[16]	2018	API calls	100 malicious and benign Java samples from Contagio malware dump	GRU	Implemented a framework used to classify android applications either benign or malware leveraging GRU and achieved 91.42% detection accuracy.	The framework execution is limited for just labelled data.
[17]	2018	code analysis (i.e., permissions and API calls)	5000 Malicious from AMD and Drebin, 2000 Benign from APKpure and APPsapk.	CNN	Presented a lightweight 1D-CNN scheme to evaluate the raw APK files for sophisticated malware and achieves 97% average accuracy.	Limited set of raw APK files may not achieve good results.
[18]	2018	API calls, Permissions	Drebin-5560 Malicious, 123453 Benign	DNN	Utilized Deep Neural Network (DNN) to classify Android malicious applications and obtained 97% classification accuracy.	The approach is susceptible to different obfuscation attacks.
[19]	2018	API sequence calls	Drebin, Genome, Virusshare, Contagio, minidump	CNN	Presented MalDozer, employing ANN to determine malware sequence from raw data of different API calls.	Limited family attribution.
[20]	2018	API calls, permissions, activities	6965 Android APK's from Drebin	DBN	proposed the image texture-based classification mechanism leveraging DBN achieves 95.6% detection accuracy.	limited feature set do not lead classifier towards high accuracy.
[21]	2018	Permissions, API calls, Intent filter	5546 Malicious from Drebin, 5224 Benign from Chinese third-party market	CNN	Presented DeepClassifyDroid employed three-step approach: feature extraction, feature embedding and CNN-based malware detection framework and achieved 97.4% detection accuracy.	Susceptible to impersonate attack.
[22]	2018	Permissions, Linux system calls	2104 Benign, 2104 Malicious from Drebin and Marvin	DNN	Proposed DDefender technique for malware identification executed on user's device to detect Android malware and achieved 95% detection accuracy.	Tool used for events capturing (Monkey Tool) is not able to capture all events generated by malwares.
[23]	2018	Permissions, API calls	1400 Malicious from Drebin, 1400 Benign from Google play store	DBN	Executed DBN to build automatic malware classifier and achieved 95% accuracy.	Susceptible to adversarial attack, not efcient for real time deployment.

TABLE 1. *(Continued.)* **Selected previous studies for android malware detection.**

[24]	2018	Permissions events generated through Monkey Tool	Running an emulator and collecting feature vectors in real-time	LSTM	Android malware detection scheme leveraging LSTM is proposed for static and dynamic analysis and achieves 97.5% and 93.4% detection accuracy respectively.	For dynamic analysis, applications were running in emulator for a short time.
[25]	2018	Android bytecode	110440 applications from virus-share, Google play, Massvet	LSTM	Proposed Deep-Refiner leveraging LSTM on the semantic structure of Android bytecode to detect Android malware and achieves 97.74% detection accuracy.	Frequent update required for labelled features and computationally complex process.
[26]	2018	Extract features through conversion of Android APk's into RGB color code	829356 applications collected through leopard mobile Inc	CNN	Proposed framework for automatic feature learning through translating applications into RGB color image and achieves 97.25% detection accuracy.	Samples are not in enough numbers to get desirable results.
[27]	2018	Requested permissions, components and filtered intents	Drebin and Virus-share	CNN, RNN, DAE, DBN, LSTM	The proposed scheme practiced different DL classifiers to efficiently detect malware in Android applications and achieves 93.6% accuracy	Computationally complex process due to large dataset of more than 1 million APK's.
[28]	2018	Permissions, API calls	Virus-Share, Malgenome Project, Google Play App Store, Virus-Total	CNN	Proposed framework utilized multimodal CNN to reflect the properties of Android applications as benign or malicious.	Prone to obfuscation technique.
[29]	2017	System Call Sequences	Android Virtual Device (AVD) system calls	CNN	Designed end-to-end malware identification technique by considering system call sequences as text and acquired 93.16% accuracy.	Not performance efficient.
[30]	2017	system calls behavior	216 malicious from contagio project, 278 applications normal from Google Play Store	ANN, DT, RF, k-NN, Gradient boosting trees, SVM	Proposed conventional malware detection scheme based on ANN and 6 diverse machine learning classifiers and achieves 97.16% for SVM.	The small number of instances are reason for not achieving efficient performance from deep learning classifier.
[31]	2017	API calls of smali code	2500 Malicious and 2500 Benign from Comodo cloud security	DBN, SAE	The author executed DBN and SAE for newly unknown Android malware detection and obtained 95.98% detection accuracy.	Susceptible to adversarial attack.
[32]	2017	Static data flow analysis	8000 Malicious from virusshare and Genome, 3000 Benign from Google play store	DBN	proposed Flow-Droid framework for malware identification leverages data flow analysis and achieved 95.05% classification accuracy.	Susceptible to adversarial attack.
[33]	2017	Opcode Sequence	27377 applications from Genome, McAfee Labs	CNN	Designed detection technique for malware relies to extract and learn features from raw data automatically and the framework achieves 98% detection accuracy.	Trained on large dataset, still not very satisfactory performance of classifier.
[34]	2017	random adversarial samples	14,679 malware variants, 17,399 Benign from MNIST CIFAR-10	DNN	Implemented adversary resistant technique that obstructs adversary from constructing impactful adversarial samples and achieves 95.2% detection accuracy.	Need to investigate on different applications.
[35]	2017	Permissions	Cyber Security Data Mining Competition (CDMC2016)	LSTM	Proposed framework converts Android permissions into features by using LSTM and achieves 89.7% accuracy.	Due to low achieved accuracy value, framework cannot be deployed in real time scenario.

called gates that regulate and even circulate the flow of information. The gates help to learn the GRU cell which information is important to store or erase. Thus, the important information is passed further to make predictions. Forget gate and input gates are also joined together to design an update gate z_t. The update gate is responsible for maintaining the amount of previous memory and new information to be hold. x_t is a current input vector and h_{t-1} is basically value calculated from previous adjacent layer. However, w_z is learnable weight matrix for update gate.

$$z_t = \sigma(w_z.[h_{t-1}, x_t]) \tag{1}$$

GRU also combines current input with previous memory at reset gate r_t. Moreover, r_t is responsible for determining how exactly the equation combines previous state and new output.

$$r_t = \sigma(w_r.[h_{t-1}, x_t]) \tag{2}$$

Tanh is a hyperbolic tangent function. The output range for tanh is $(-1,1)$. Moreover, h_t is the calculated value for current cell.

$$h_t = \tanh(r_t * [h_{t-1}, x_t]) \tag{3}$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * h_t \tag{4}$$

The architecture of GRU is simple than standard RNN but proved to be performance and speed efficient. The basic architecture of GRU can be seen in Fig.1.

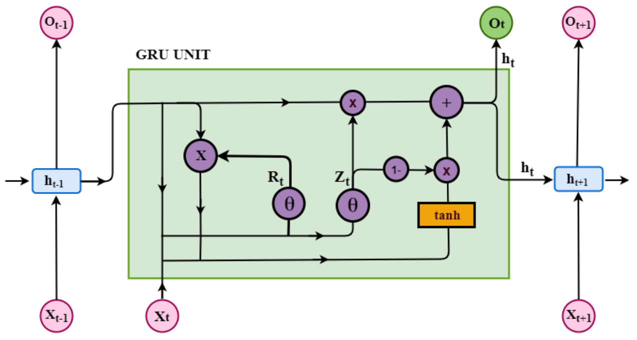

FIGURE 1. The Architecture of basic Gated Recurrent Unit (GRU).

B. LONG SHORT-TERM MEMORY

Long short-term memory (LSTM) [36], a variant of Recurrent Neural Network (RNN) is an incredible classifier for temporal data mining and learning. LSTM model utilize extraordinary module known as constant error carousel (CEC) to spread constant error signal through time to learn long-term features and dependencies. Furthermore, through utilization of well-designed "gate" structure prevents backpropagated error. The internal values of CEC is decided by the "gate" state according to current information and previous flow to control the data steam and memory. One LSTM cell is comprised of three gates and two states named input gate, forget gate, output gate, hidden state and cell state respectively. Given an input sequence x = $\{x_1 + x_2, \ldots., x_t\}$ where input gate, forget gate, and output gate in LSTM structure, individually are documented as i_t, f_t and o_t and the weights attached to them are W_i, W_f, W_o, b_i, b_f and b_o. For each time step, LSTM updates two states, hidden state h_t and cell state c_t and σ symbol is used for sigmoid function. The basic and cell architecture of LSTM is illustrated in Fig.2. In equation 5,

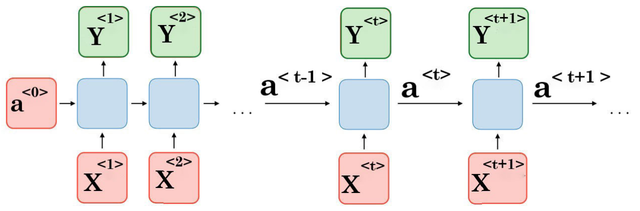

FIGURE 2. The Architectural diagram of a basic LSTM Unit.

The f_t gate decides how much information is to be kept in the cell. h_{t-1} is obtained output value from previous layer while x_t is the current input vector. b_f is referred as forget gate bias and W_f is weight matrix.

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{5}$$

After the decision of information to be kept, next step is to update cell state and it is achieved through input gate i_t.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{6}$$

Tanh which is a hyperbolic tangent function, generates a vector of new candidate values, c_t.

$$c_t = \tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{7}$$

The current candidate value which decides to change the value is done through multiplication of old value and f_t. Further, i_t* c_t is added to the equation.

$$c_t = f_t * c_{t-1} + i_t * c_t \tag{8}$$

Finally, a filtered output o_t of the cell state is obtained as a final output.

$$o_t = \sigma(W_o * [h_t - 1, x_t] + b_o) \tag{9}$$
$$h_t = o_t * \tanh(c_t) \tag{10}$$

For a typical LSTM cell, the sequence data is considered as an input to LSTM cell and hidden layers are fully connected to the input layer. The size of LSTM output layer depends upon to the number of classes to classify.

C. CONVOLUTIONAL NEURAL NETWORK

After achieving outstanding results in the fields of image recognition, speech recognition, computer vision, and natural language processing, Convolutional Neural Network is now prospering its roots in cyber security. CNN can learn the important features automatically compared to conventional feature selection algorithms. CNN [37] is considered as a sequence of interconnected processing components intended to transform the set of inputs to the set of required outputs in Fig.4. Input, output and hidden layers are main components of a CNN classifier. CNN performs multiple operations on the input data, which include convolution, pooling, flattening and padding, finally the network relates to a fully connected neural network.

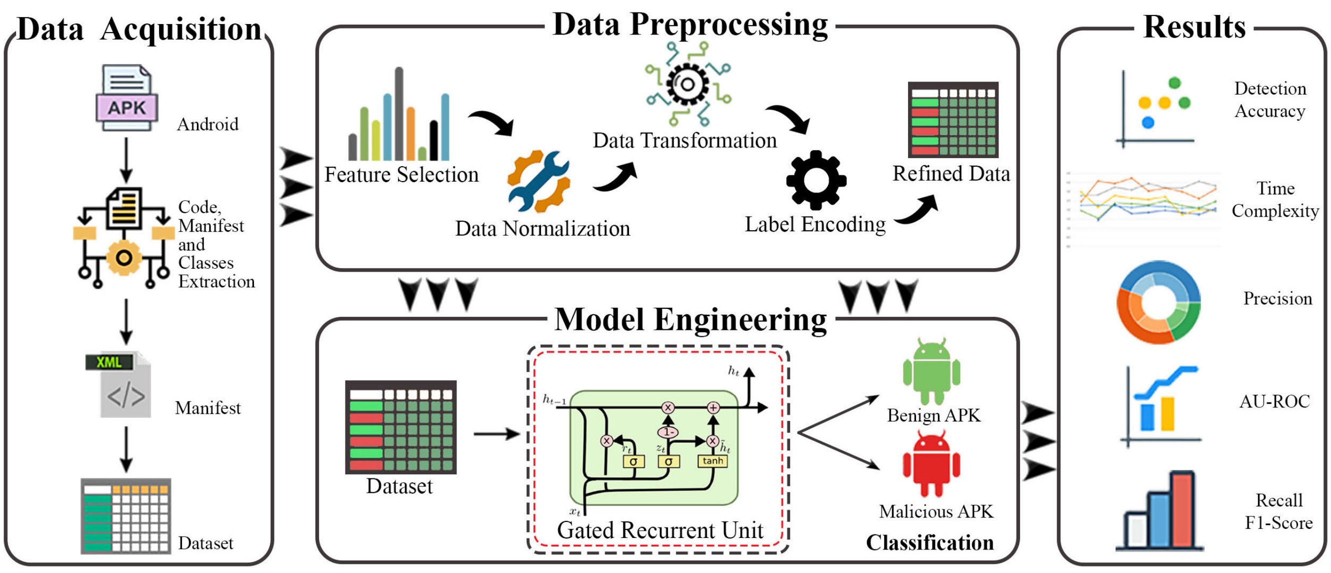

FIGURE 3. The Simplified Overview of Proposed GRU-based Android Malware Detection scheme.

D. DEEP NEURAL NETWORK

Deep Neural Network (DNN) [38] is recognised as standard Artificial Neural Network with multiple inter-connected layers between input and output layers. The DNN finds the correct mathematical calculation to convert the input into the output. The single layer of DNN contains several neurons where computations are performed. The node receives input, performs operations while utilizing stored weights, applies activation function and then finally pass the information to the next subsequent node until it reaches to a conclusion. Fig.5 depicts complete simplified overview of DNN.

IV. METHODOLOGY

Fig.3 illustrates the simplified overview of the proposed graphical processing unit (GPU) empowered GRU-based deep learning scheme for detection of Android malware. The first Data Acquisition phase is structured to generate dataset which includes with code, manifest file and classes extraction from Android package kit (Apk) and the feature vector generation from manifest file. The data preprocessing phase is comprised of feature selection, data normalization, data transformation and label encoding to convert dataset into a classifiers acceptable format. In the model engineering phase DL-driven GPU-accelerated GRU model is employed for detection of ever evolving sophisticated cyber threats and attacks in Android ecosystem. Finally, the achieved results have been rigorously evaluated through standard performance evaluation metrics (e.g., Accuracy, Precision, Recall, F1-score, AU-ROC and etc).

A. DATA ACQUISITION

An Android APK includes with a manifest file named as *AndroidManifest.xml*. Manifest file contains metadata that supports the access privileges, installation and execution of

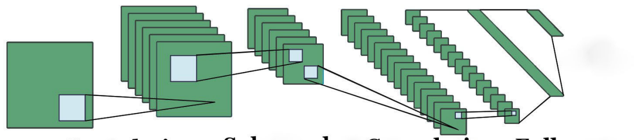

FIGURE 4. The Basic Architecture of Convolutional Neural Network.

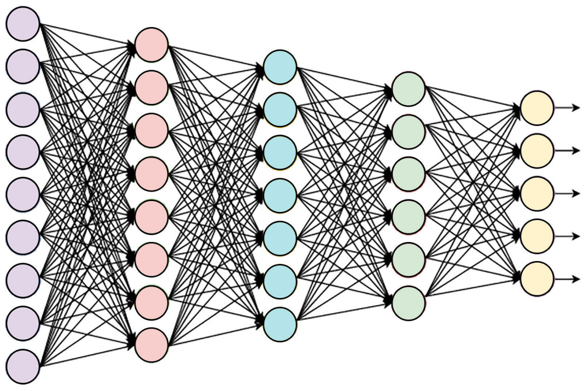

FIGURE 5. The Basic Architecture of Deep Neural Network.

the Android packages. In the proposed methodology, Java code is structured to extract manifest file from an Android APK. For the extraction of useful information and feature vector generation from manifest file, python script has been written. The designed code extensively examines manifest file and extracts diverse features (i.e., permissions, API calls and filtered intents). The initial extracted features through python script includes with permissions like access to camera, GPS, Mic or touchscreen. Mostly malicious intended application asks to access those components which are not

TABLE 2. Description of proposed gated recurrent unit (GRU) architecture for malware detection system.

Algorithms	*GRU*		*LSTM*		*DNN*		*CNN*	
	Layer	**Neurons**	**Layer**	**Neurons**	**Layer**	**Neurons**	**Layer**	**Neurons**
	GRU	400	*LSTM*	400	*Dense*	100	CNN	150
	GRU	350	*LSTM*	250	*Dense*	75	*CNN*	350
	GRU	300	*LSTM*	200	*Dense*	50	*CNN*	300
	GRU	125	*LSTM*	125	*Dense*	32	*CNN*	125
	GRU	50	*LSTM*	50	*Dense*	3	*CNN*	50
	Dense	32	*Dense*	32			*Dense*	30
	Dense	3	*Dense*	3			*Dense*	3
Activation function	*Relu, Softmax*							
Loss function	categorical crossentropy							
Optimizer	*Adam*							
Epochs	30							
Batch-size	32							
Learning rate	0.01							

even required for any purpose or the functionality of the application.

Requested permissions are also considered important for Android malware detection system as they are granted to the application at installation time. We nearly accumulated one thousand distinct permissions that are used by various Android applications. An Android application is built upon four main components known as Activity, Service, Broadcast Receiver and Content provider. These four components are also considered as features of dataset. Intents are a message-passing mechanism within the application's components or with other connected applications. Filtered intents are also considered.

B. DATA PREPROCESSING

Data preprocessing refers to a process of all the transformations on raw data before input to an algorithm for efficient performance. Furthermore, it also reduces the complexity of the available resources in term of storage and time. Apk's of AMD and Androzoo are combined to generate dataset for our proposed Android malware detection. It satisfies every one of the criteria such as labelled dataset, complete manifest files features, malware diversity and heterogeneity. Through python script, around 19000 features were extracted but 150 most appearing features has been selected. To increase the effectiveness of malware detection system there is a need to shift all the feature values into the scaled version. For this, python Standard Scaler function is used to perform the normalization of the dataset. To reduce data redundancy and improve data integrity data transformation has been performed which remove duplicate, null and infinity values. The conversion of each value in a column to a number is also executed through label encoding.

C. MODEL ENGINEERING

The DL-driven GPU-empowered GRU is employed for detection of ever evolving sophisticated cyber threats and attacks in Android. An efficient, robust and scalable Android malware detection scheme have been implemented which detect multi-class malwares like Backdoor and Trojan. The complete design of the proposed benchmark GPU-empowered GRU classifier and other experimented architectures for performance evaluation (i.e., *Layers*, *Neurons*, *Activation Functions*, *Loss Function*, *Optimizer*, *Batch Size*, *Epochs* and *Learning rate*) are detailed in Table.2.

D. DATASET

The selection of current state of the art dataset plays a significant role in analysing the performance of a malware detection system. For malware detection, dataset comprised of 38842 APK's (i.e., 30831 benign APK's have been collected from Androzoo [39] and 8011 Malware APK's from Android Malware Dataset (AMD) [40]. AMD contains 10 diverse classes of malware (i.e., Backdoor, Trojan, Trojan-Banker, Trojan-clicker, Trojan-Dropper, Trojan-SMS, Trojan-Spy, Adware, HackerTool, Ransom) with 71 distinct malware families. For dataset, we merge 6 different classes of trojan into single trojan class. Thus, the complete distribution of dataset is across 3 different classes including Benign, Backdoor and Trojan which can also be seen in Fig.6.

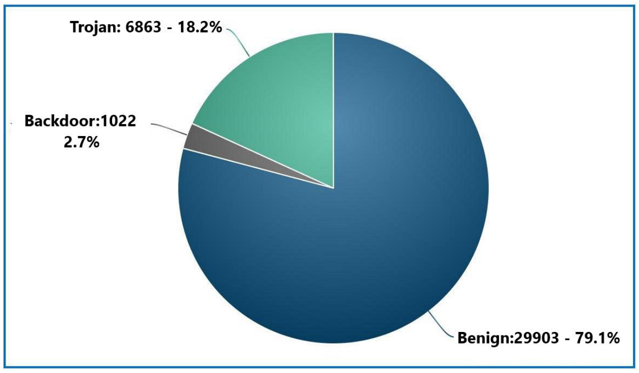

FIGURE 6. The Dataset Distribution Graph for Proposed scheme.

V. EXPERIMENTAL SETUP, RESULTS AND DISCUSSION

The section provides complete overview of the experimental setup, standard performance evaluation metrics and results along with discussion.

A. EXPERIMENTAL SETUP

For the experimental setup, the authors have employed Intel processor, Graphical Processing Unit (GPU) for hardware setup. Whereas, the proposed methodology is implemented using Keras for software implementation [41]. The python language is utilized for development of proposed methodology. Detailed hardware and software specifications can be seen in Table.3.

TABLE 3. Hardware and software specifications for proposed GRU-based android malware detection architecture.

	Processor	Model	Generation
CPU	Core-i9(4.0Ghz)	9900K	Nineth(9th)
GPU	NVIDIA GTX-1080		
OS	Windows 10 - 64 Bit		
RAM	32GB - 3600 MHz		
language	Python		
Software	Numpy, Tensorflow, Scikitlearn, Pandas		

B. EVALUATION METRICS

For comprehensive performance evaluation of proposed malware detection model, we have employed standard evaluation metrics (i.e., accuracy, precision, recall, F1-score, and ROC curve). For a more thorough evaluation, we have also calculated extended evaluation metrics (i.e., True Negative Rate, (TNR), Matthews Correlation Coefficient (MCC), Negative Predictive Value (NPV), False Positive Rate (FPR), False Negative Rate (FNR), False Discovery Rate (FDR) and False Omission Rate (FOR). The detailed description of performance evaluation metrics can be seen in [42].

The True Positve, True Negative, False Positive and False Negative are described as:

True Positive (TP) is equal to number of malicious records classified as malicious.

True Negative (TN) is equal to number of benign samples predicted as benign.

False Positive (FP) is equal to number of benign samples mis-classified as malicious one.

False Negative (FN) is equal to number of malicious records mis-classified as benign.

1) ACCURACY

Accuracy is the number to correctly predicted samples over total number of samples.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

2) PRECISION

Precision is the ratio of malicious applications that are classified correctly, to the total number of all malicious applications.

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

3) RECALL

Recall is the number of correctly predicted values from total records for each class. Recall is also called True Positive Rate (TPR).

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

4) F1-SCORE

F1 score shows the correlation between recall and precision.

$$F1\text{-score} = \frac{2 * TP}{2 * TP + FP + FN} \tag{14}$$

5) CONFUSION MATRIX

A confusion matrix is used to describe the overall performance of a classification model. Confusion Matrix is represented in either for binary or multi-class. The Confusion Matrix is useful to measure the values of accuracy, precision, recall and AUC-ROC curve.

6) AUC-ROC

AUC-ROC represents the degree of separability and mainly represents the performance of multi-class classification problems. This one of the most important evaluation metrics to distinguish between multiple classes. The Higher AUC, the model is good at predicting 0s as 0s, 1s as 1s. The plot of ROC curve is between TP rate and FP rate.

C. DISCUSSION

The Cuda-empowered GRU-based Android malware detection scheme is proposed for 3 diverse classes (i.e., benign, trojan and backdoor).

For comprehensive performance evaluation, we compare the proposed technique with our constructed contemporary DL algorithms such as Long short-term memory (LSTM), Convolutional Neural Network (CNN) and Deep Neural Network (DNN). The confusion matrix in Fig.7 depicts the performance efficiency of our proposed technique for multi attack classification. The highly achieved values of True Positive (TP) and True Negative (TN) defines the performance of our system to be effectively utilized for ever-evolving sophisticated cyber threats and malware detection in Android.

To measure, how accurately the Android applications are classified as benign or malicious, detection accuracy, precision, recall and F1-score is also calculated and depicted in Fig.8. The proposed GPU-enabled GRU-based malware detection technique achieves 98.96% detection accuracy, 99.38% precision, 99.31% recall and 99.35% F1-score for malware classification. However, DNN accomplished 99.59% precision value which is higher than our proposed model is due to less number of records and less complex architecture. classified the relevant instances more correctly.

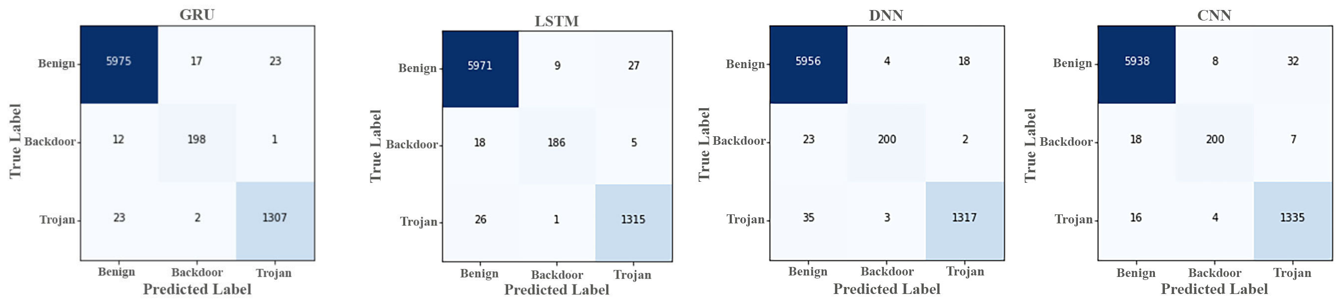

FIGURE 7. Confusion Matrices of proposed GRU, LSTM, DNN and CNN model.

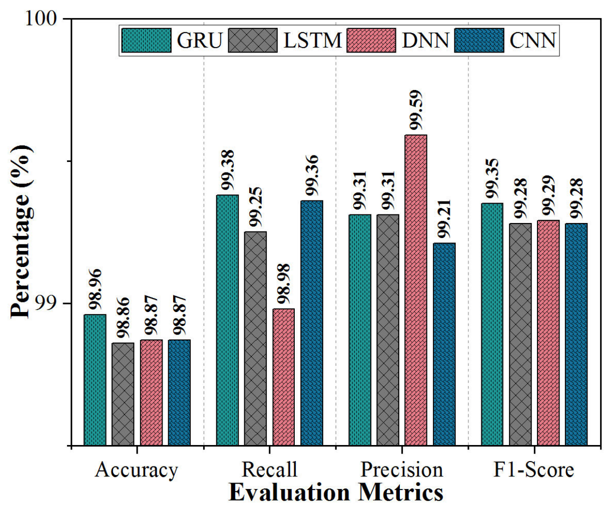

FIGURE 8. Accuracy, Precision, Recall, F1-Score values for proposed GRU-based Android malware detection technique.

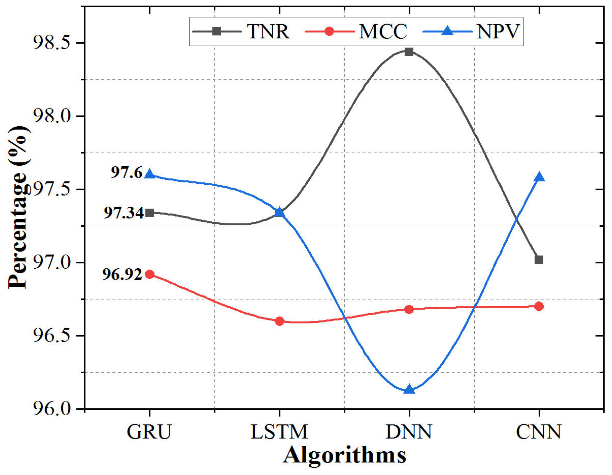

FIGURE 9. TPR, MCC, NPV values for proposed cuda-empowered GRU-based model.

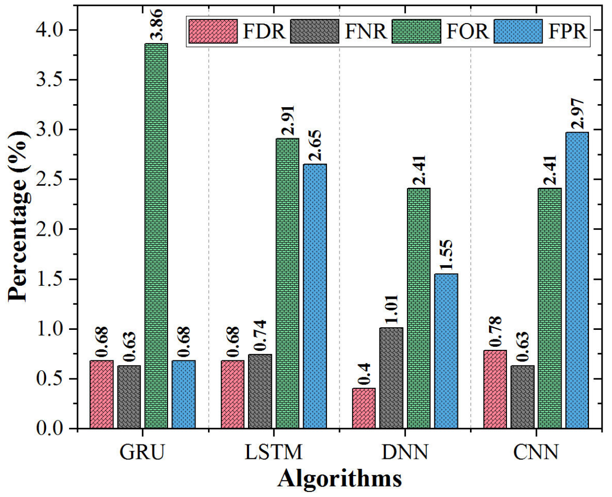

FIGURE 10. The achieved FDR, FNR, FOR and FPR values for Proposed cuda enabled GRU-based technique.

Our proposed technique is overall achieving high performance compared to other constructed classifiers. To rigorously evaluate the proposed model, True Negative Rate (TNR), Matthews Correlation Coefficient (MCC) and Negative predicted values (NPV) are also calculated. our proposed mechanism achieves 97.34%, 96.92%, 97.60% for TNR, MCC and NPV as presented in Fig.9. The proposed mechanism clearly outperforms the rest of the classifiers.

Moreover, False Negative Rate (FNR), False Positive Rate (FPR), False Omission Rate (FOR) and False Discovery Rate (FDR) are fundamentally determined to show the classifier execution for mischaracterization of instances. The proposed GRU-based malware detection solution accomplished 0.68%, 0.63%, 2.4% and 0.68% for FDR, FNR, FOR and FPR respectively as shown in Fig.10. Results for FDR, FOR, FNR and FPR clearly show great performance of our proposed architecture.

The AU-ROC illustrates the relationship between True Positive Rate (TPR) and False Positive Rate (TNR). The area under the curve shows robust performance of the proposed technique as shown in Fig.11. AU-ROC's clearly demonstrates the reliable performance of our proposed technique.

The time efficiency of our proposed technique is defined in Fig.12. The proposed GRU model took 1005 (milliseconds) almost equal to 1 second for 7560 instances in testing

TABLE 4. Comparison of proposed GPU-empowered GRU technique with other contemporary existing state-of-the-art solutions for android malware detection.

Papers	Algorithms	Multi-class	Accuracy(%)	Precision(%)	Recall(%)	F1-score(%)	Detection Time
Proposed Technique	GRU	✓	98.96	99.31	99.38	99.35	1005(ms)
Jha et al. [16]	GRU	-	91.42%	-	-	-	-
Lee et al. [12]	GRU, CNN	-	97.70%	-	-	-	40(ms)
Stuart et al. [11]	DAN	-	97.30	98.00	96.60	97.30	-
Wei et al. [43]	DAE-CNN	-	98.60	98.65	98.65	98.69	-
karbab et al. [19]	DNN	-	90.00	98.00	99.00	99.00	-
kim et al. [28]	DNN	-	98.00	98.00	99.00	99.00	-

Abbreviation Terms: GRU – Gated Recurrent Unit, DAN – Discriminative Adversarial Network, DAE-CNN – Deep Auto-Encoder- Convolutional Neural Network, DNN – Deep Neural Network.

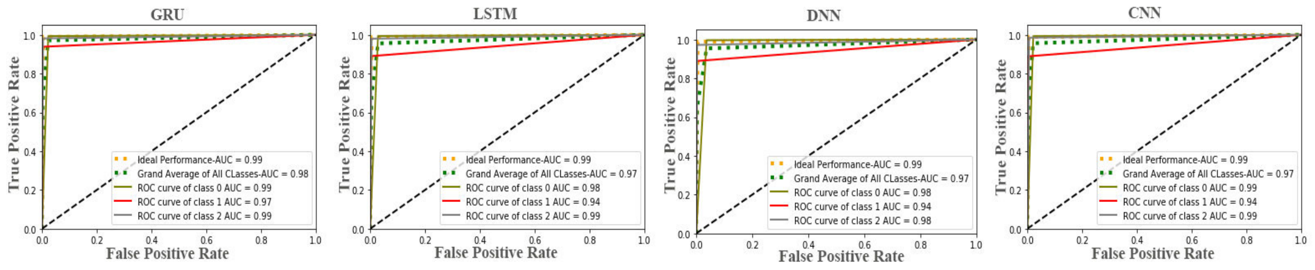

FIGURE 11. The graphical representation of AU-ROC Curve for proposed GRU and other experimented classifiers like LSTM, DNN and CNN.

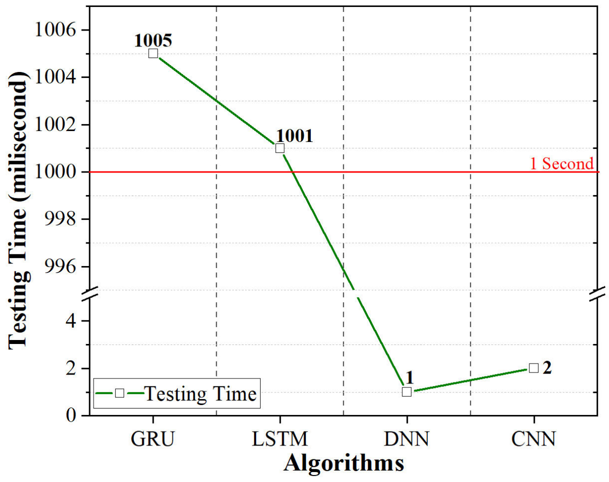

FIGURE 12. Testing Time of Proposed GPU-accelerated GRU, LSTM, DNN and CNN.

phase. The GRU is clearly showing a trivial trade off of speed efficiency with other compared algorithms. Though, the speed efficiency of GRU is not very promising, however; there is a room for improvement in terms of time complexity. Our future work plans to improve the speed efficiency of the proposed algorithm.

Consequently, the proposed technique is also compared with current benchmarks. The comprehensive comparison of GRU-based technique with current state of the art is shown in Table.4. The achieved values evidently show the outperformance of our proposed mechanism for Android malware detection.

VI. CONCLUSION

Increasing demand of open and prevalent environment of Android OS not only revolutionized the digital landscape but also bring novel sophisticated cyber security vulnerabilities, threats and attacks. To tackle sophisticated multi-class malware threats and attacks, we propose a robust, scalable and efficient DL-based Android malware detection technique. The proposed mechanism has been thoroughly evaluated with standard performance metrics and compared extensively with current benchmarks and our constructed contemporary DL-driven algorithms. The proposed scheme outperforms in terms of high detection accuracy that means accurately identifying prevalent varied Android malwares. The efficient and timely detection of the proposed technique can remarkably help for subsequent mitigation and prevention of attacks. Finally, we endorse varied deep learning architectures to find a promising solution to combat novel and emerging sophisticated Android malwares.

REFERENCES

[1] L. Goasduff. (2020). *Gartner Says Global Smartphone Sales Fell Slightly in the Fourth Quarter of 2019*. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2020-03-03-gartner-s%ays-global-smartphone-sales-fell-slightly-in#:~:text=Global%20sales%20of%20%20smartphones%20to,%2C%20smartp-hone%20sales%20declined%201%25

[2] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 3rd Quart., 2019.

[3] Statista. (2019). *Number of Smartphone Users Worldwide From 2016 to 2021 (in Billions)*. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-w

[4] A. Qamar, A. Karim, and V. Chang, "Mobile malware attacks: Review, taxonomy & future directions," *Future Gener. Comput. Syst.*, vol. 97, pp. 887–909, Aug. 2019.

[5] S. Sen, E. Aydogan, and A. I. Aysan, "Coevolution of mobile malware and anti-malware," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 10, pp. 2563–2574, Oct. 2018.

[6] L. Liu, O. De Vel, Q.-L. Han, J. Zhang, and Y. Xiang, "Detecting and preventing cyber insider threats: A survey," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1397–1417, 2nd Quart., 2018.

[7] M. Conti, Q. Q. Li, A. Maragno, and R. Spolaor, "The dark side(-channel) of mobile devices: A survey on network traffic analysis," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2658–2713, 4th Quart., 2018.

[8] P. Yan and Z. Yan, "A survey on dynamic mobile malware detection," *Softw. Qual. J.*, vol. 26, no. 3, pp. 891–919, Sep. 2018.

[9] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, "Significant permission identification for machine-learning-based Android malware detection," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3216–3225, Jul. 2018.

[10] S. Ni, Q. Qian, and R. Zhang, "Malware identification using visualization images and deep learning," *Comput. Secur.*, vol. 77, pp. 871–885, Aug. 2018.

[11] S. Millar, N. McLaughlin, J. Martinez del Rincon, P. Miller, and Z. Zhao, "DANdroid: A multi-view discriminative adversarial network for obfuscated Android malware detection," in *Proc. 10th ACM Conf. Data Appl. Secur. Privacy*, Mar. 2020, pp. 353–364.

[12] W. Y. Lee, J. Saxe, and R. Harang, "Seqdroid: Obfuscated Android malware detection using stacked convolutional and recurrent neural networks," in *Deep Learning Applications for Cyber Security*. Springer, 2019, pp. 197–210.

[13] J. Booz, J. McGiff, W. G. Hatcher, W. Yu, J. Nguyen, and C. Lu, "Towards deep learning-based approach for detecting Android malware," *Int. J. Softw. Innov.*, vol. 7, no. 4, pp. 1–24, Oct. 2019.

[14] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-droid: Deep learning based Android malware detection using real devices," *Comput. Secur.*, vol. 89, Feb. 2020, Art. no. 101663.

[15] X. Xiao, S. Zhang, F. Mercaldo, G. Hu, and A. K. Sangaiah, "Android malware detection based on system call sequences and LSTM," *Multimedia Tools Appl.*, vol. 78, no. 4, pp. 3979–3999, Feb. 2019.

[16] P. K. Jha, P. Shankar, V. Sujadevi, and P. Prabhaharan, "Deepmal4j: Java malware detection employing deep learning," in *Proc. Int. Symp. Secur. Comput. Commun.*, Springer, 2018, pp. 389–402.

[17] C. Hasegawa and H. Iyatomi, "One-dimensional convolutional neural networks for Android malware detection," in *Proc. IEEE 14th Int. Colloq. Signal Process. Appl. (CSPA)*, Mar. 2018, pp. 99–102.

[18] D. Li, Z. Wang, and Y. Xue, "Fine-grained Android malware detection based on deep learning," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, May 2018, pp. 1–2.

[19] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "MalDozer: Automatic framework for Android malware detection using deep learning," *Digit. Invest.*, vol. 24, pp. S48–S59, Mar. 2018.

[20] L. Shiqi, T. Shengwei, Y. Long, Y. Jiong, and S. Hua, "Android malicious code classification using deep belief network," *KSII Trans. Internet Inf. Syst.*, vol. 12, no. 1, 2018.

[21] Y. Zhang, Y. Yang, and X. Wang, "A novel Android malware detection approach based on convolutional neural network," in *Proc. 2nd Int. Conf. Cryptogr., Secur. Privacy ICCSP*, 2018, pp. 144–149.

[22] H. Alshahrani, H. Mansourt, S. Thorn, A. Alshehri, A. Alzahrani, and H. Fu, "DDefender: Android application threat detection using static and dynamic analysis," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2018, pp. 1–6.

[23] W. Li, Z. Wang, J. Cai, and S. Cheng, "An Android malware detection approach using weight-adjusted deep learning," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Mar. 2018, pp. 437–441.

[24] R. Vinayakumar, K. P. Soman, P. Poornachandran, and S. Sachin Kumar, "Detecting Android malware using long short-term memory (LSTM)," *J. Intell. Fuzzy Syst.*, vol. 34, no. 3, pp. 1277–1288, Mar. 2018.

[25] K. Xu, Y. Li, R. H. Deng, and K. Chen, "DeepRefiner: Multi-layer Android malware detection system applying deep neural networks," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Apr. 2018, pp. 473–487.

[26] T. H.-D. Huang and H.-Y. Kao, "R2-D2: ColoR-inspired convolutional NeuRal network (CNN)-based AndroiD malware detections," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 2633–2642.

[27] M. Nauman, T. A. Tanveer, S. Khan, and T. A. Syed, "Deep neural architectures for large scale Android malware analysis," *Cluster Comput.*, vol. 21, no. 1, pp. 569–588, Mar. 2018.

[28] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for Android malware detection using various features," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 3, pp. 773–788, Mar. 2019.

[29] H. Liang, Y. Song, and D. Xiao, "An end-to-end model for Android malware detection," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Jul. 2017, pp. 140–142.

[30] L. Singh and M. Hofmann, "Dynamic behavior analysis of Android applications for malware detection," in *Proc. Int. Conf. Intell. Commun. Comput. Techn. (ICCT)*, Dec. 2017, pp. 1–7.

[31] S. Hou, A. Saas, L. Chen, Y. Ye, and T. Bourlai, "Deep neural networks for automatic Android malware detection," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, Jul. 2017, pp. 803–810.

[32] D. Zhu, H. Jin, Y. Yang, D. Wu, and W. Chen, "DeepFlow: Deep learning-based malware detection by mining Android application for abnormal usage of sensitive data," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2017, pp. 438–443.

[33] N. McLaughlin, J. M. del Rincon, B. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doupé, "Deep Android malware detection," in *Proc. 7th ACM Conf. Data Appl. Secur. Privacy*, 2017, pp. 301–308.

[34] Q. Wang, W. Guo, K. Zhang, A. G. Ororbia, X. Xing, X. Liu, and C. L. Giles, "Adversary resistant deep neural networks with an application to malware detection," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 1145–1153.

[35] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Deep Android malware detection and classification," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1677–1683.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] M. Sabokrou, M. Fayyaz, M. Fathy, Z. Moayed, and R. Klette, "Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes," *Comput. Vis. Image Understand.*, vol. 172, pp. 88–97, Jul. 2018.

[38] S. Walczak, "Artificial neural networks," in *Advanced Methodologies and Technologies in Artificial Intelligence, Computer Simulation and Human-Computer Interaction*. Hershey, PA, USA: IGI Global, 2019, pp. 40–53.

[39] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "AndroZoo: Collecting millions of Android apps for the research community," in *Proc. 13th Int. Workshop Mining Softw. Repositories MSR*, 2016, pp. 468–471, doi: 10.1145/2901739.2903508.

[40] F. Wei, Y. Li, S. Roy, X. Ou, and W. Zhou, "Deep ground truth analysis of current Android malware," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*. Springer, 2017, pp. 252–276.

[41] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. Newton, MA, USA: O'Reilly Media, 2019.

[42] A. Tharwat, "Classification assessment methods: A detailed tutorial," *Appl. Comput. Informat.*, to be published.

[43] W. Wang, M. Zhao, and J. Wang, "Effective Android malware detection with a hybrid model based on deep autoencoder and convolutional neural network," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 8, pp. 3035–3043, Aug. 2019.

IRAM BIBI received the B.S. degree (Hons.) in software engineering from the National University of Modern Languages (NUML), Islamabad, Pakistan, in 2017, and the Master of Science degree in information security from COMSATS University, Islamabad, in 2020. She is currently working with ProSanct, for one year, as a Research Assistant. Her research interests include analysis and detection of network based cyber threat and attacks for Android, the Internet of Things, and software defined networking.

ADNAN AKHUNZADA is an enthusiastic and dedicated professional with extensive 12 years of Research and Development experience in ICT industry and academia, with demonstrated history and a proven track record of high impact published research (i.e., Patents, Journals, Transactions, Commercial Products, Book chapters, Reputable Magazines, Conferences, and Conference Proceedings). His experience as an Educator and a Researcher is diverse. It includes work as a Lecturer, a Senior Lecturer, a year Tutor, an Occasional Lecturer at other engineering departments, as an Assistant Professor with COMSATS University Islamabad (CUI), a Senior Researcher with RISE SICs, Vasteras, AB, Sweden, as a Research Fellow and the Scientific Lead at DTU Compute, The Technical University of Denmark (DTU), and a Visiting Professor having mentorship of graduate students, and supervision of academic and Research and Development projects at UG and PG level. He has also been involved in international accreditation such as Accreditation Board for Engineering and Technology (ABET), and curriculum development according to the guidelines of ACM/IEEE. He is currently involved in various EU and Swedish funded projects of cyber security. His main research interests include cyber security, machine learning, deep learning, reinforcement learning, artificial intelligence, blockchain and data mining, information systems, large scale distributed systems (i.e., edge, fog, and cloud, SDNs), the IoT, Industry 4.0, and the Internet of Everything (IoE). He is a member of the technical programme committee of varied reputable conferences and editorial boards. He has been serving as an Associate Editor for IEEE ACCESS.

JAHANZAIB MALIK received the B.S. degree (Hons.) in software engineering from the National University of Modern Languages, Islamabad, Pakistan, and the M.Sc. degree in information security from COMSATS University, Islamabad. He is currently a Researcher with the National Cyber Security Auditing and Evaluation Laboratory (NCSAEL), National University of Science and Technology (NUST), Islamabad. His research interests include software defined networking, smart devices security, threat detection and intelligence, malware analysis and detection, application of deep learning and machine learning in cyber defense, distributed computing, and big data.

JAVED IQBAL received the Ph.D. degree in computer science from the University of Malaya, Malaysia, in 2016. He is currently an Assistant Professor with the Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan. His research interests include software process improvement, requirements engineering, and software development outsourcing. He received the Award and the Adolph Lomb Medal (OSA).

ARSLAN MUSSADDIQ received the B.S. degree in electrical engineering (telecommunication) from Bahria University, Islamabad, Pakistan, in 2011, and the M.S. degree in communication and network engineering from University Putra Malaysia, in 2015. He is currently pursuing the Ph.D. degree with the Department of Information and Communication Engineering, College of Engineering, Yeungnam University, Gyeongsan, South Korea. His research interests include wireless networking, the Internet of Things, wireless resource management, routing protocols, and ad hoc networks. He was a recipient of the Outstanding Dissertation (M. S. level) Award at the IEEE Malaysia Communication Society and the Vehicular Technology Society Joint Chapter, in 2015.

SUNGWON KIM received the B.S. and M.S. degrees from the Department of Control and Instrumentation Engineering, Seoul National University, South Korea, in 1990 and 1992, respectively, and the Ph.D. degree from the School of Electrical Engineering and Computer Sciences, Seoul National University, in 2002. From 1992 to 2001, he was a Researcher with the Research and Development Center, LG Electronics, South Korea. From 2001 to 2003, he was a Researcher with the Research and Development Center, AL Tech, South Korea. From 2003 to 2005, he was a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, USA. In 2005, he joined the Department of Information and Communication Engineering, Yeungnam University, Gyeongsan, South Korea, where he is currently a Professor. His research interests include resource management, wireless networks, mobile networks, performance evaluation, and embedded systems.