Received June 28, 2019, accepted August 13, 2019, date of publication August 26, 2019, date of current version September 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2937347

Cyber Security Threats Detection in Internet of Things Using Deep Learning Approach

FARHAN ULLAH[1,2], HAMAD NAEEM[1], SOHAIL JABBAR[3], SHEHZAD KHALID[4], MUHAMMAD AHSAN LATIF[5], FADI AL-TURJMAN[6], (Member, IEEE), AND LEONARDO MOSTARDA[7], (Member, IEEE)

[1]College of Computer Science, Sichuan University, Chengdu 610065, China
[2]Department of Computer Science, COMSATS University Islamabad, Sahiwal Campus, Sahiwal 57000, Pakistan
[3]Department of Computer Science, National Textile University, Faisalabad 38000, Pakistan
[4]Department of Computer Engineering, Bahria University, Islamabad 44000, Pakistan
[5]Department of Computer Science, University of Agriculture Faisalabad, Faisalabad 38000, Pakistan
[6]Department of Computer Engineering, Antalya Bilim University, 07010 Antalya, Turkey
[7]Computer Science Department, Camerino University, 62032 Camerino, Italy

Corresponding author: Sohail Jabbar (sjabbar.reserarch@gmail.com)

ABSTRACT The IoT (Internet of Things) connect systems, applications, data storage, and services that may be a new gateway for cyber-attacks as they continuously offer services in the organization. Currently, software piracy and malware attacks are high risks to compromise the security of IoT. These threats may steal important information that causes economic and reputational damages. In this paper, we have proposed a combined deep learning approach to detect the pirated software and malware-infected files across the IoT network. The TensorFlow deep neural network is proposed to identify pirated software using source code plagiarism. The tokenization and weighting feature methods are used to filter the noisy data and further, to zoom the importance of each token in terms of source code plagiarism. Then, the deep learning approach is used to detect source code plagiarism. The dataset is collected from Google Code Jam (GCJ) to investigate software piracy. Apart from this, the deep convolutional neural network is used to detect malicious infections in IoT network through color image visualization. The malware samples are obtained from Maling dataset for experimentation. The experimental results indicate that the classification performance of the proposed solution to measure the cybersecurity threats in IoT are better than the state of the art methods.

INDEX TERMS Internet of Things, data mining, cyber security, software piracy, malware detection.

I. INTRODUCTION

IoT is the interconnection of physical moving objects ''Things'' through internet embedded with an electronic chip, sensors, and other forms of hardware. Each device is uniquely identified globally by Radio Frequency Identifier (RFID) tags. These smart objects communicate with Other connected nodes and can be monitored and controlled remotely [1]. IoT offers pervasive connectivity to a broad range of intelligent physical objects, service industries, cloud computing services, and applications. IBM stated that the number of connected devices through the internet is expected to increase up to 50 billion by 2020 [2]. It will increase the number of communication networks with the connection of smart objects as well as the amount of big data that may be shared using cloud infrastructure. The IoT enabled technologies can be used to develop smart cities, education system, e-shopping, e-banking, maintain our health, manage industry, and to entertain and protect human beings [3]. The IoT devices can be used for an open attack due to always available on the network. The industrial IoT-cloud can be easily targeted by malware infection and pirated software for harmful usage and to compromise security [4], [5]. The software piracy is the development of software by reusing source codes illegally from someone else's work and disguise as the original version. The cracker may copy the logic of the original software by reverse engineering procedures and then design the same logic in another type of source codes [6]. It is a severe threat to internet security, which gives access to unlimited downloads of pirated software, open-source codes and, promotes

The associate editor coordinating the review of this article and approving it for publication was Sherali Zeadally.

and advertises of pirated versions. It rapidly increases each year and gives substantial economic loss to the software industry [7]. The Business Software Alliance (BSA) 2016 report stated that the public software piracy ratio is approximately 39%, which results in business damages up to 52.2$ billion every year. Many types of research have shown that every software contains plagiarized source codes in the context of logic with a range of 5% to 20% [8], [9]. The intelligent software plagiarism techniques are required to catch the plagiarized source code in pirated software. Different plagiarism detection techniques are proposed, i.e., clone detection, source code similarity identification, software bugs analysis, and software birthmark investigation [10], [11]. These techniques mainly are structure and text-based analysis. The structure-based technique examines the basic structure of source codes, syntax trees, graph behavior, and function call graph of subroutines. Because of this, it is limited to a specific programming language structure. So, if a cracker reuses the logic of the original software to another type of programming language, then, it is hard to catch due to different structure behavior. The industrial IoT cloud services may be used to secure and protect smart devices by designing intelligent software plagiarism and malware detection techniques.

Currently, malicious attacks are more easily abrupt due to the growing number of IoT networks. The malware attacks are usually planned to infect the privacy of IoT nodes, computer systems, and smartphones over the internet. Several scanning techniques are proposed to detect windows-based malware by using specific signatures. The malware identification analysis is divided into two main methods, i.e. static and dynamic approaches. In dynamic approach, malware patterns are learned while executing code in a real-time virtual environment. Malicious behavior can be observed by function calls, function parameters' exploration, data flow, instruction traces, and visual investigation of codes. There are automated online tools available that can be used to investigate the dynamic behavior of malicious codes. i.e., CW Sandbox, Anubis, and TT analyzer. This method is more time consuming due to monitoring every dynamic behavior of source codes [12]–[14]. The static malware analysis methods do not need the real-time execution of source codes. It may be used to capture the layout information of malware binaries. The signature-based malware identification techniques are static based, i.e., control flow graph, opcode frequency, n-gram, and string signature. The disassembling tools, i.e., IDA Pro and OllyDbg [15], are applied to uncover the executables before implementing static based algorithms. These disassemblers are used to extract the hidden patterns from binary executables. Then, these patterns are used to retrieve encoded string from executables. The byte sequence technique is a static based analysis that can be used to extract n-bytes sequences from these patterns. The functional call graph is a static analysis method used to extract the structural analysis of codes [16].

A. CHALLENGES

1) SOFTWARE PIRACY

Currently, every third installed software application is pirated. The intellectual digital property and authorship rights of software are not physical due to globally accessible on the internet and thus, hard to sustain [17]. The attacker may crack the original software and re-design the logic into another type of programming language. It is challenging to catch the crackers' malicious activities in cross-domain source codes because each programming language has different syntax and semantic structures. There are some tools available that can translate one type of source code to another type, i.e., MoHCA-Java [6], [18]. The widely available open-source projects made it easy for the cracker to copy the original idea and design their software. One cannot get paid by his thoughts, but for the ability to offer the solutions into real-world.

2) MALWARE DETECTION

The conventional methods may solve code obfuscation concerns, but high computational cost is needed regarding texture feature mining using malware visualizations. These types of feature extraction procedures do not perform well with extensive malware data analysis. Currently, malware is continuously generating, update, and manipulate that makes the detection more challenging [19], [20]. The proposed malware detection method tries to respond to the following queries:

- How to identify malware with reduced overhead?
- How to extract malware features with less computational cost?
- How to process big malware datasets to get better accuracy?

In this paper, we proposed a combined deep learning approach to identify the pirated and Malware attacks on industrial IoT cloud. The TensorFlow deep neural network is designed to capture the pirated software using source code plagiarism. Further, the deep convolution neural network is designed to capture the malicious patterns of malware through binary visualization. The combined solutions of the proposed approach are much promising in terms of classification performance. The main contributions of the proposed work are:

- Malware threat detection with less computational cost
- Large scale malware detection with better accuracy
- Software Similarity identification from different programmers
- detection of a pirated copy of original software

The remaining paper is organized follows. Section II contains a literature review, section III includes the proposed methodology, the results and discussions are given in section IV, and finally, the conclusion is given in section V.

II. LITERATURE REVIEW

For efficient threat detection in the IoT environment, more studies are conducted on malware detection and software plagiarism detection to accomplish high Identification performance and reduce time cost.

Several studies have observed the influence of different features of software plagiarism detection. Most of the state of the art work is done on a single programming language. It means that, if a cracker alters the control flow of source code to other data structure in the similar programming language, then, the present literature is valuable. In [21], the software benchmark was used to detect a threat in java source codes. It extracted the control flow of source codes to retrieve structural features. The benchmarks of two source codes were compared to compute the similarity. In [22], the author proposed a hybrid approach to obtain code similarity from intermediate code generation using compiler level features.

Additionally, the unsupervised learning approach is used to measure similarity in source codes. The similar functionalities of different source codes were used to detect plagiarism. In [23], a source forager search engine approach is used to mine similar features between C and C++ code in response to user questions. This search engine is further used to extract different properties from code and proceeds in the shape of k functionalities present in the codes. The logic shows the control flow of source code, and it can be used for software similarity. In [24], the logic-based approach is proposed to compute the behavior of dissimilarity in two source codes. If there is no dissimilarity, then, it initiated in plagiarism problem. The symbolic execution and preconditions reasoning was used to capture the semantics for dissimilarities from execution paths. In [25], the Latent Semantic Analysis (LSA) is used to identify plagiarism in students' assignments. The LSA is combined with PlaGate to examine semantic resemblance between source codes fragments. Apart from this, the author described how different code chunks were significant concerning semantics. The syntax tree is used to capture the abstract and syntactic view of source code. A syntax tree from any source code is based on the parse tree. It may be used to compare different source codes based on syntactic structures. In [26], the parse tree kernel technique is used to excerpt similar source codes fragments between java files. The proposed method did not give better results because of unbalanced variants of nodes in essential characteristics. The fingerprinting technique is proposed to excerpt the similarity between source codes.

Several studies have conducted static, dynamic, hybrid, and visualization analysis for malware detection in the past. The detailed summary of each type of analysis is presented. Static analysis techniques usually consist of features extraction by static means from binary files using binary data extraction tools. In [27], uses a sequence of variable length instructions and classify worms from the binaries of benign files to classify them based on machine learning. They classified a dataset consists of 1444 worm files, and 1330 benign files and results showed that classification accuracy of 96% [28] proposed an approach based on N-Opcode sequences and applied machine learning using SVM classifier. They attained an accuracy of 98% in malware detection. Opcodes are the machine code mnemonics, whereas the cosine similarity and critical instruction sequence techniques were used for the malware identification process. Results showed that every version of similar malware family share some common core signatures and can be captured using core API calls sequences. It [29] proposed an obfuscation scheme to check the limitations of static analysis approaches. The experimental results showed that the static analysis independently is not sufficient for useful analysis of malware samples. Static analysis approaches can easily be avoided if the malware is packed or obfuscated. Therefore, some robust behavioral features are required in the analysis. Such types of malware detection approaches are usually called dynamic analysis approaches. Dynamic analysis approaches are usually based on the execution of binary samples in a controlled environment to extract features within a virtual machine. In [30], the author proposed a technique to automate the manual process of malware analysis. They identify the malicious behavior within the program, which was not previously shown by a benign application. This approach gives a short amount of information regarding malicious behavior. Therefore, it can provide significant insights for understanding malware. In [31], the author proposed a useful clustering-based approach which can automatically scale a considerable amount of malware samples into group/classes of clusters based on their execution behavior. They also extended the Anubis system for additional network analysis and tainted tracking to generate automated truce report for the selected malware samples. Their extended system was able to characterize different activities of the program more abstractly. Hybrid approaches are introduced to tackle issues of time, computational cost, and to improve malware detection systems. In [32] collectively used both static and dynamic features extracted from malware samples to train a malware classifier and named their approach OPEM. They considered static features as frequency occurrences of operational codes. But, dynamic features may be execution traces of executable files and system calls. The results showed that the hybrid approach performs much better as a combined approach rather than running of static and dynamic methods separately. Many studies have been directed for visualization of malware features to improve the performance of classification results as well as a reduction in time, size, and resource overhead. For example, [33] introduced a CNN and image-based malware classification method. This approach achieved 98.52% classification accuracy. It randomly separated 10% samples for testing from malware family. Reference [34] designed a deep learning model for malware detection. The proposed approach achieved 98% classification accuracy for 9339 malware samples. In [35], the CNN model is proposed for malware classification. The proposed approach got 94.5% classification accuracy.

III. PROPOSED METHODOLOGY

In this study, we propose an architecture model for cybersecurity threats and protection measures in industrial IoT,

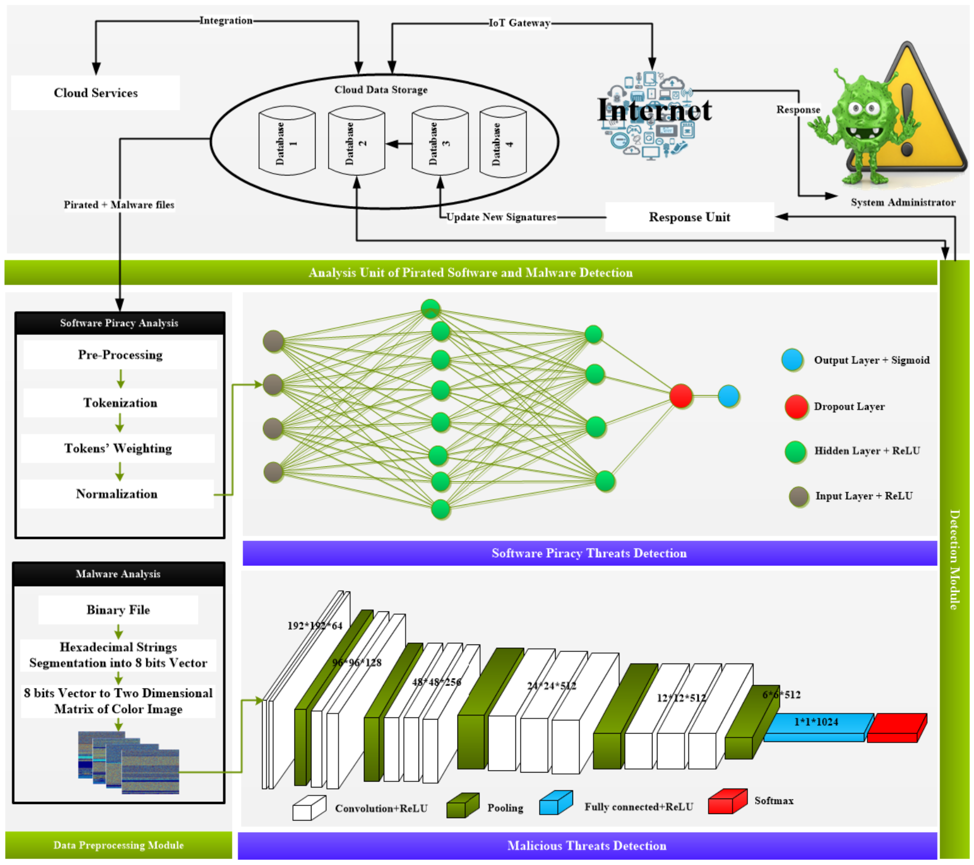

FIGURE 1. Architecture model for cyber security threats prediction in IoT.

as shown in Figure 1. Four databases are implemented in cloud storage to process the malware binaries and software pirated files. The raw network traffic data is stored in database 1, and the second database comprises a list of earlier malware data. Further, the third database is used to store new signatures of detected malware attacks. The cracker stores pirated software in database 4 through IoT devices. It works as a storage place for pirated copies where cracker tries to spread these copies by IoT network. This massive amount of data needs high processing, i.e. time and cost. The first database sent raw data to the pre-processing module. It preprocesses the raw data and captures useful features. Pre-processed data is further submitted to the detection module. The detection module captures malware and pirated software attacks by learning from the signatures in databases two and four. If any malicious activity is observed in the network, then, the proposed system warns the administrator for suitable action.

A. MALWARE THREAT DETECTION

1) DATA PREPROCESSING

The color images are generated from raw binary files to transform the malware detection problem using image classification problem. It differentiates the proposed research from state of the art methods [19], [35], [36], in which malware binary files convert into a grayscale image with 256 colors. This method does not depend on any reverse engineering tool such as disassembler and decompile. The color images can retrieve better features as compared to grayscale images with only 256 colors.

Further, the better features of malware images can outperform in the classification of malware families. Before, numerous malware detection methods based on machine learning algorithms [19], [20] offered better outcomes using grayscale images. The color images are transformed into grayscale visualization, and then feature extraction techniques were used to classify malware type. The classification performance is improved using feature reduction methods to decrease the features' set. The results showed that machine learning algorithms are not a better choice for malware detection because it generates exponential values using color images. The deep learning algorithms outperform with big malware datasets as these types of methods can use filters to decrease noise automatically. Thus, the use of color images generates better results using deep learning techniques.

The conversion of malware binary file to color image comprises of four phases. First, the hexadecimal strings (0-15) are produced from raw binary files. Second, a hexadecimal stream is divided into a chunk of the 8-bit vector where each 8-bit segment is measured as an unsigned integer (0-255). Third, the 8-bit vector is subsequently converted into a two-dimensional matrix space. Fourth, each 8-bit integer generates from two-dimensional space is plotted with of red, green, and blue shaded colors. Figure 1 presents the complete phases of data preprocessing section.

2) DEEP CONVOLUTIONAL NEURAL NETWOR

The Deep Convolutional Neural Network (DCNN) is proposed to conduct in-depth malware data analysis. The DCNN contains five modules, as shown in Figure 1. The input layer is used to receive training images for the designed neural network model. First, the convolution layer is used to decrease the noise and give better signal characteristics. Convolutional kernel width, the number of hidden units and learning rate are optimized to get maximum accuracy of the proposed deep learning model. Second, pooling layer is used to decrease the data overhead preserving useful information. Third, a fully connected layer is used to convert the two-dimensional array into the one-dimensional and then input it to the specific classifier. Fourth, malware families from respective images are identified by using the classifier.

3) CONVOLUTION LAYER

The essential features are extracted by reducing the image of parameters using convolution layer. Convolution layer, i.e. interpretation invariance, rotation invariance, and scale invariance. It decreases the over-fitting problem and gives the generalization concept to the main architecture. The input of the convolutional layer is several maps, as shown in equation 1 [37].

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

where M_j denotes the cluster of given maps; k_{ij}^l defines convolution kernel, used for joining the ith input feature map with jth output feature map; b_j^l is the bias consistent to the ith feature map, and is the activation function.

4) POOLING LAYER

The sub-sampling layer term is generally used for pooling layer which offers two modes of pooling, i.e., maximum and average pooling. It is not disturbed by backward propagation and used to minimize the consequence of image distortion. It also decreases the features' factor and increases the proposed DCNN functioning, as shown in equation 2.

$$x_j^l = f\left(down\left(x_j^{l-1}\right) + b_j^l\right) \tag{2}$$

where down (.) performs a pooling task, and b represents bias value.

a: FULLY CONNECTED LAYER

The fully connected layer is used to classify the output of pooling layer. Every neuron links with the preceding neuron with a corresponding fully connected layer. This layer is used to improves the model generalization competency by decreasing the overfitting issue. In fine-tuning, filters are applied to the original image for reducing noise. The number and size of kernels are specified to provide better signal characteristics.

b: LEARNING

The malware samples are accurately categorized with the respective family name. In this research, we use Softmax-Cross-Entropy loss to train the proposed DCNN model. The Loss for the training data k is shown in equation 3.

$$Loss = -\log\left(\frac{\exp(f_{zt})}{\sum_k \exp(f_{zt})}\right) \tag{3}$$

where fzt is the rank for the kth class; and fzt is the score for the correct family. The parameters of the model are learned using Adam optimizer, which tries to minimize the loss incurred on the training data.

B. SOFTWARE PIRACY THREAT DETECTION

The main goal of the proposed deep learning methodology is to catch the pirated software from different types of source codes. A deep learning methodology is designed to detect plagiarism in various types of source codes. The plagiarized version of the software is the pirated copy in which cracker used the logic of the original software, as shown in Figure 1. First, the source codes are tokenized in preprocessing steps to reduce the dimensions of the data and extract meaningful features for next step. The TensorFlow framework with Keras API deep learning algorithm is applied on extracted significant features to catch the plagiarism between different types of source codes. The dataset[1] is collected from GCJ, which includes 400 different source codes documents from 100 programmers. The dataset is collected from Google Code Jam (GCJ) database.

1) PRE-PROCESSING AND FEATURE EXTRACTION

The detection of pirated software is among different types of source codes is a challenging task because each source code has different syntax and semantic structures. We used software plagiarism methods to identify source code similarity. The preprocessing techniques are used to break the source codes in small pieces for deep analysis. It includes stemming, root words and frequency extraction and stops words removal. It converts the codes into meaningful information and removes noisy data. The data is cleaned used from undesirable details, i.e., special symbol, constants, stop words. Then, the tokenization process is used to transform the cleaned data into useful tokens. The stemming, root words and frequency constraints are used to mine more valuable features in the preprocessing phase. Then, the weighting techniques are used to zoom the contribution of each token. The TFIDF and Logarithm of Term Frequency (LogTF) are used in the weighting phase [38], [39]. Mathematically TFIDF is

[1]https://github.com/Jur1cek/gcj-dataset

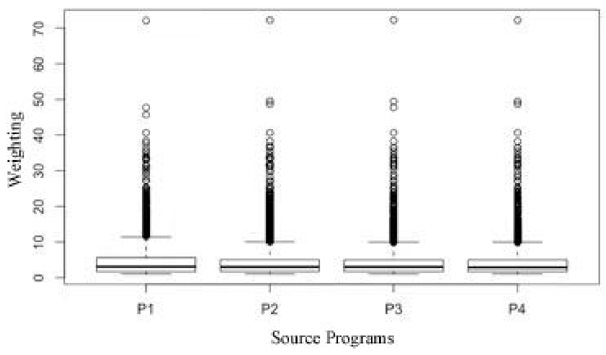

FIGURE 2. Box plot of source codes weighting values.

defined as in Equation 4.

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D) \tag{4}$$

where t denotes token, f denotes the number of frequency, d denote every individual document, and D denotes all documents used in the dataset.

2) DEEP LEARNING WITH TENSORFLOW FRAMEWOKR

The TensorFlow is a machine learning system which is used for high-level computations in a complex environment. We can implement different types of machine and deep learning algorithms by calling specific Application Programing Interface (API) of TensorFlow. It has different types of layers which can be configured for complex computations, training the data, and supervise the state-run of each function [40], [41]. The in-depth learning approach is designed to identify similar source codes in different types of programming languages using TensorFlow framework. Then, the extracted similar codes are used to determine the pirated software. The weighting values are used as input to the deep learning model. The dense layer also called fully connected layer, which is configured for input and output data. There are three dense layers configured with 100, 50, and 30 neurons, respectively. The first layer is used to receive the data with an input variable with an input shape parameter. Every neuron is receiving information from previous layers and thus densely connected. The 4th dense layer is used for the output variable to target the plagiarized code.

The deep learning approach is enhanced using drop out layer in the context of activation and loss function, optimizer, and learning error rate. The overfitting problem is also solved using dropout layer. The rectifier (ReLU) activation method is used for input variables to get the patterns of received data [42]. Mathematically, it is expressed as the positive chunk of its argument given in equation 5.

$$f(x) = x^{+} = \max(0, x) \tag{5}$$

where x denotes the input to the equivalent neurons. The sigmoid is an effective logistic method used to grip the multi-class problem [43]. It is defined mathematically as in equation 6.

$$S(x) = \frac{1}{1+e^{-x}} \tag{6}$$

where S represents the sigmoid function. The Adam optimizer also called stochastic descent gradient, is used in compiling and optimizing deep learning model. It calculates the discrete adaptive learning rates for each limitation [44], [45]. The decaying means of pas squared gradients are shown in Equation 7 and 8.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{7}$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g^2 \tag{8}$$

where m$_t$ and v$_t$ are the predictable means of the first and second instant gradients, respectively. Adam optimization algorithm works on these formulas to calculate the average of predecessor and successor moments. These are estimates which are used to update running exponential average of first gradient m$_t$ and square gradient v$_t$.

Tensorflow based algorithm has the following contributions:

- It integrates various types of computational APIs to design and extend machine learning approaches for large scales of data, i.e. GitHub framework
- This framework trains the model automatically based on input, hidden and output layers with different activation functions.
- It offers reliable services while updating and extending the designed model.
- These types of algorithms can be configured and run from small devices to big networks.

TABLE 1. Comparison with state of the art piracy detection techniques.

Proposed by	Source Code	Technique	Accuracy (%)
Bandara, Wijayrathna [46]	Java	KNN	86%
Cosma, G. and M. Joy [25]	Java	LSA	99%
Son, J.-W [26]	Java	Parse Tree	90%
Ullah, F [47]	C++, Java	LSA	80%
Ullah, F [6]	C++, Java, Python	MLR	86%
Our Proposed Approach	**C++, Java, Python**	**Deep Neural Network**	**96%**

IV. RESULTS AND DISCUSSIONS

We investigate the proposed research in case study 1 and case study 2.

A. PERFORMANCE EVALUATION OF CASE STUDY 1

The software plagiarism measure may be used to investigate the code similarity in pirated software. We took source code

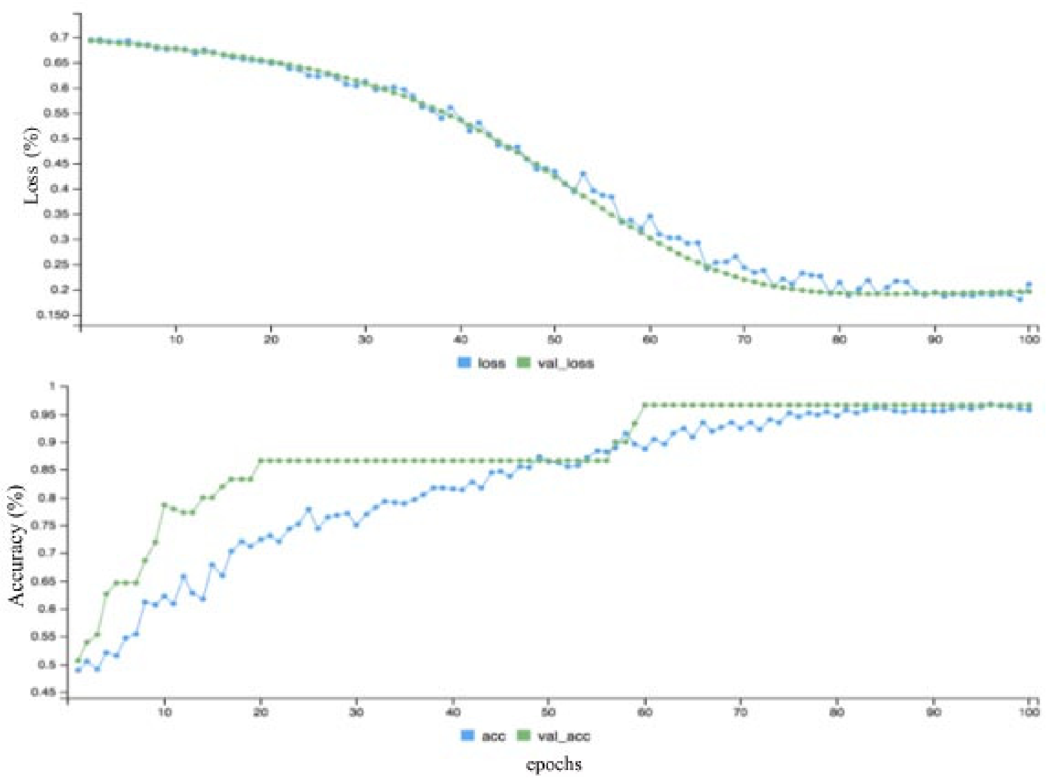

FIGURE 3. The dynamic plot of loss, validation loss, accuracy and validation accuracy for source codes.

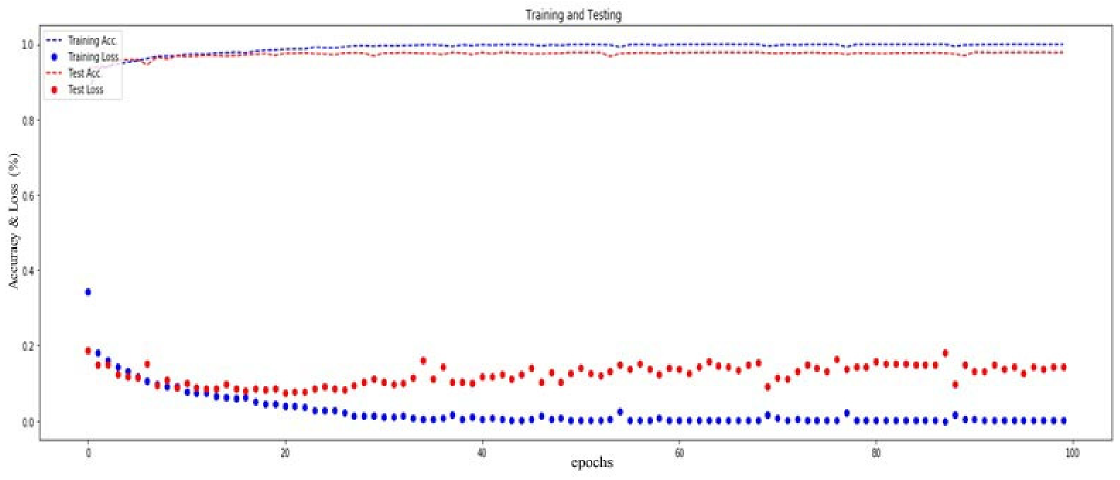

FIGURE 4. Dynamic visualization of accuracy validated accuracy, loss and validated loss (Image Ratio: 224*224).

dataset from Google Code Jam (GCJ) to analyze the proposed methodology for software piracy. First, the dataset is preprocessed to get the useful tokens for each source with frequency details. The preprocessing process includes stemming, root word, maximum and minimum token's length, maximum and minimum token's frequency, etc. Second, features selection and extraction techniques, i.e., Term Frequency and Inverse Document Frequency (TFIDF) and Logarithm Term Frequency (LogTF) are used to retrieve the tokens' weighting. The weighting methods are used to zoom the contribution of each token in the document or across all documents. The box plot of weighting values is shown in figure 2. The P1, P2, P3, and P4 on x-axis denote the programming solutions of four questions, respectively. The weighting values of each source code are given on the y-axis. Each programmer solved four different programming problems, so, we have four input variables in the first dense layer. The second and third are configured as hidden layers

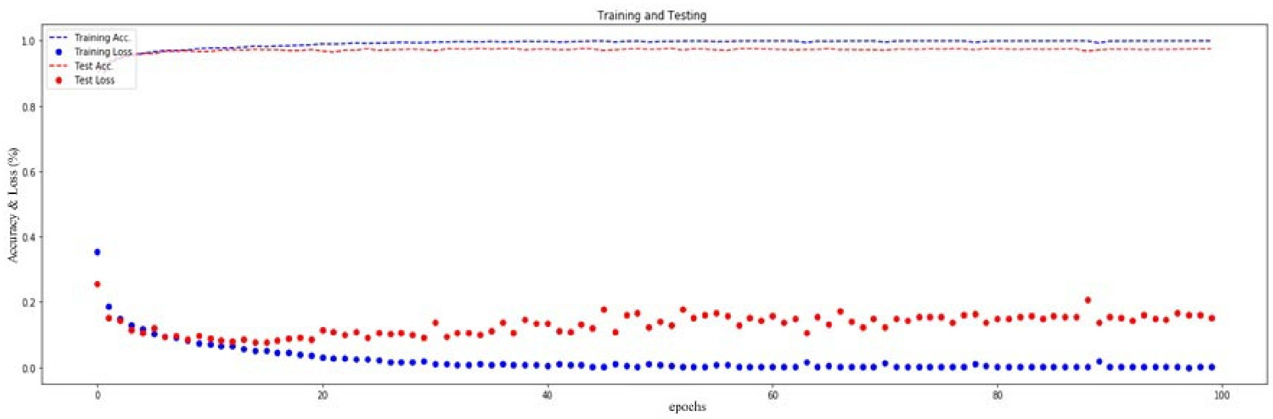

FIGURE 5. Dynamic graph of accuracy validated accuracy and loss and validated loss (Image Ratio: 229*229).

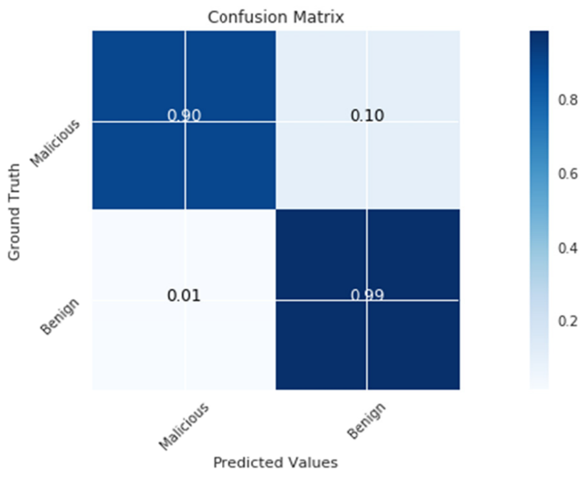

FIGURE 6. Confusion matrix for 229*229 image ratio.

to improve accuracy. The dropout layer is configured with input and each hidden layer to overcome the overfitting problem.

Further, in fine-tuning number of neurons in different layers with activation function and learning error rate also increase the classification accuracy. The dynamic visualization of accuracy, validation accuracy, loss, and validation loss in percentage measure are given in Figure 3. In the upper panel, the blue curve shows the loss, and the green curve shows validation loss. Initially, both curves start from same 0.7 points, and up to 35 epochs behave the same, but after the blue curve fluctuating on the green curve. As both curves decreasing so, it has fewer chances of overfitting problem. If these curves are increasing or behave opposite, then overfitting occurs. The comparison is made of the proposed approach with another state of the art techniques, as shown in Table 1.

B. PERFORMANCE EVALUATION OF CASE STUDY 2

The effect is measured for the different number of malware image ratios in terms of classification performance of the proposed approach. The image ratios are taken 224 × 224 and 229 × 229, respectively. We selected 14,733 malware and 2486 benign samples from the Leopard Mobile dataset.[2] The 229×229 ratio gives better accuracy as compared to 224×224 ratio regarding classification performance. However, a significant difference between 229 × 229 and 224 × 224 image sizes is classification accuracy. Thus, it is concluded that 229 × 229 image ratio is a worthy selection for the proposed malware detection approach. For 224 × 224 and 229 × 229 image ratios, the dynamic graph for accuracy, validated accuracy, loss, and validated loss are shown in Figure 3 and 4. To validate the importance of image ratio for better accuracy, table 2 shows a brief comparison of the classification results of different image ratios. 229 × 229 image ratio obtained 97.46% testing accuracy with 36s computational time for the Leopard Mobile dataset. The confusion matrix for 229 × 229 image ratio is shown in Figure 5. We also compared the performance of proposed deep learning-based malware detection with existing machine learning-based malware detection researches [35], [47] and [48] on the Leopard Mobile dataset. These approaches used traditional image feature extraction descriptors, namely, GIST, LBP, and CLGM descriptors with SVM classifier. The highest classification accuracy got by the proposed approach was 97.46%, GIST + SVM was 86.1%, LBP + SVM was 78.05%, and CLGM + SVM was 92.06% respectively. The F-measure is a weighted average of precision and recall. It is the harmonic mean of precision and recall. The highest F-measure obtained by the proposed method was 97.44%, GIST+SVM was 85.82%, LBP+SVM was 77.49%, and CLGM + SVM was 91.98% respectively. Further, figure 6 shows the confusion matrix for 229 × 229 image ratio. The ground truth means true label in a dataset and predicted values shows predicted classes after applying the proposed deep learning algorithm. There are two classes, i.e. malicious and benign used in malware dataset. Overall, 90% of classes are predicted for malicious files, with 10% miss classifications error. Similarly, 99% of benign classes

[2]https://sites.google.com/site/nckuikm/home

TABLE 2. Comparison of the proposed malware detection approach based on different image ratios.

Image Ratio	Precision (%)	Recall (%)	F-measure(%)	Accuracy (%)	Time (s)
224×224	95.16	95.10	95.13	95.10	20s
229×229	97.43	97.46	97.44	97.46	36s

Note: NS=Not specified, CA=Classification accuracy, PR=Precision rate, RR=Recall rate

TABLE 3. Comparision of malware detection approach with other methods based on classification accuracy.

Works	**Year**	**Samples (Malware/Benign)**	**Technique**	**F-measure (%)**	**CA (%)**
GIST+SVM [17]	2011	4000/2000	ML	85.82	86.1
LBP+SVM [46]	2018	4000/2000	ML	77.49	78.05
CLGM+SVM [47]	2019	4000/2000	ML	91.98	92.06
Our Proposed Approach	**-**	**14,733/2486**	**DL**	**97.44**	**97.46**

are predicted with a 1% miss classification. Table 3 shows that the proposed malware detection method outperforms as compared to three existing machine learning-based works in terms of classification performance.

V. CONCLUSION

The industrial IoT based network is rapidly growing in the coming future. The detection of software piracy and malware threats are the main challenges in the field of cybersecurity using IoT-based big data. We proposed a combined deep learning-based approach for the identification of pirated and malware files. First, the TensorFlow neural network is proposed to detect the pirated features of original software using software plagiarism. We collected 100 programmers' source codes files from GCJ to investigate the proposed approach. The source code is preprocessed to clean from noise and to capture further the high-quality features which include useful tokens. Then, TFIDF and LogTF weighting techniques are used to zoom the contribution of each feature in terms of source code similarity. The weighting values are then used as input to the designed deep learning approach. Secondly, we proposed a novel methodology based on convolution neural network and color image visualization to detect malware using IoT. We have converted the malware files into color images to get better malware visualized features. Then, we passed these visualized features of malware into deep convolution neural network. The experimental results show that the combined approach retrieve maximum classification results as compared to the state of the art techniques. Tokenization process extracts keywords from source codes, but it does not show the internal view of source codes. The abstract syntax tree and control flow graph feature to capture the syntactic and control flow of source codes. In future, we will try to use these features for detection of pirated copies. Malware detection for unknown set of malware is a big issue. Further, we will try to propose an algorithm that can detect malware for unknown malware families.

REFERENCES

[1] C. R. Srinivasan, B. Rajesh, P. Saikalyan, K. Premsagar, and E. S. Yadav, "A review on the different types of Internet of Things (IoT)," *J. Adv. Res. Dyn. Control Syst.*, vol. 11, no. 1, pp. 154–158, 2019.

[2] G. J. Joyia, R. M. Liaqat, A. Farooq, and S. Rehman, "Internet of Medical Things (IOMT): Applications, benefits and future challenges in healthcare domain," *J. Commun.*, vol. 12, no. 4, pp. 240–247, 2017.

[3] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[4] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "Android malware detection using deep learning on API method sequences," Dec. 2017, *arXiv:1712.08996*. [Online]. Available: https://arxiv.org/abs/1712.08996

[5] S. Jabbar, K. R. Malik, M. Ahmad, O. Aldabbas, M. Asif, S. Khalid, K. Han, and S. H. Ahmed, "A methodology of real-time data fusion for localized big data analytics," *IEEE Access*, vol. 6, pp. 24510–24520, 2018.

[6] F. Ullah, J. Wang, M. Farhan, M. Habib, and S. Khalid, "Software plagiarism detection in multiprogramming languages using machine learning approach," *Concurrency Comput., Pract. Exper.*, to be published.

[7] D.-K. Chae, J. Ha, S.-W. Kim, B. Kang, and E. G. Im, "Software plagiarism detection: A graph-based approach," in *Proc. 22nd ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2013, pp. 1577–1580.

[8] Y. Akbulut and O. Dönmez, "Predictors of digital piracy among Turkish undergraduate students," *Telematics Inform.*, vol. 35, no. 5, pp. 1324–1334, 2018.

[9] M. ShanmughaSundaram and S. Subramani, "A measurement of similarity to identify identical code clones," *Int. Arab J. Inf. Technol.*, vol. 12, pp. 735–740,Dec. 2015.

[10] C. Ragkhitwetsagul, "Measuring code similarity in large-scaled code Corpora," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Oct. 2016, pp. 626–630.

[11] S. Imran, M. U. G. Khan, M. Idrees, I. Muneer, and M. M. Iqbal, "An enhanced framework for extrinsic plagiarism avoidance for research article," *Tech. J.*, vol. 23, no. 01, pp. 84–92, 2018.

[12] A. Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer, "Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey," *Inf. Secur. Tech. Rep.*, vol. 14, no. 1, pp. 16–29, 2009.

[13] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," *ACM Comput. Surv.*, vol. 44, no. 2, p. 6, Feb. 2012.

[14] I. Ghafir, J. Saleem, M. Hammoudeh, H. Faour, V. Prenosil, S. Jaf, S. Jabbar, and T. Baker, "Security threats to critical infrastructure: The human factor," *J. Supercomput.*, vol. 74, no. 10, pp. 4986–5002, Oct. 2018.

[15] I. Raz, "Introduction to reverse engineering," Cs. tau. ac. il, 2011.

[16] E. Gandotra, D. Bansal, and S. Sofat, "Malware analysis and classification: A survey," *J. Inf. Secur.*, vol. 5, no. 2, p. 56, 2014.

[17] A. Moore, *Intellectual Property and Information Control: Philosophic Foundations and Contemporary Issues*. Abingdon, U.K.: Routledge, 2017.

[18] S. Malabarba, P. Devanbu, and A. Stearns, "MoHCA-Java: A tool for C++ to Java conversion support," in *Proc. Int. Conf. Softw. Eng.*, May 1999, pp. 650–653.

[19] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: Visualization and automatic classification," in *Proc. 8th Int. Symp. Vis. Cyber Secur.*, Jul. 2011, p. 4.

[20] A. Makandar and A. Patrot, "Malware class recognition using image processing techniques," in *Proc. Int. Conf. Data Manage., Anal. Innov. (ICDMAI)*, Feb. 2017, pp. 76–80

[21] H.-I. Lim, H. Park, S. Choi, and T. Han, "A method for detecting the theft of Java programs through analysis of the control flow information," *Inf. Softw. Technol.*, vol. 51, no. 9, pp. 1338–1350, 2009.

[22] J. Yasaswi, S. Kailash, A. Chilupuri, S. Purini, and C. V. Jawahar, "Unsupervised learning based approach for plagiarism detection in programming assignments," in *Proc. 10th Innov. Softw. Eng. Conf.*, Feb. 2017, pp. 117–121.

[23] V. Kashyap, D. B. Brown, B. Liblit, D. Melski, and T. Reps, "Source forager: A search engine for similar source code," Jun. 2017, *arXiv:1706.02769*. [Online]. Available: https://arxiv.org/abs/1706.02769

[24] F. Zhang, D. Wu, P. Liu, and S. Zhu, "Program logic based software plagiarism detection," in *Proc. IEEE 25th Int. Symp. Softw. Rel. Eng.*, Nov. 2014, pp. 66–77.

[25] G. Cosma and M. Joy, "An approach to source-code plagiarism detection and investigation using latent semantic analysis," *IEEE Trans. Comput.*, vol. 61, no. 3, pp. 379–394, Mar. 2012.

[26] J.-W. Son, T.-G. Noh, H.-J. Song, and S.-B. Park, "An application for plagiarized source code detection based on a parse tree kernel," *Eng. Appl. Artif. Intell.*, vol. 26, no. 8, pp. 1911–1918, Oct. 2013.

[27] M. Siddiqui, M. C. Wang, and J. Lee, "Detecting Internet worms using data mining techniques," *J. Systemics, Cybern. Inform.*, vol. 6, no. 6, pp. 48–53, 2009.

[28] B. Kang, S. Y. Yerima, K. Mclaughlin, and S. Sezer, "N-opcode analysis for Android malware classification and categorization," in *Proc. Int. Conf. Cyber Secur. Protection Digit. Services*, Jun. 2016, pp. 1–7.

[29] A. Moser, C. Kruegel, and E. Kirda, "Limits of static analysis for malware detection," in *Proc. 23rd Annu. Comput. Secur. Appl. Conf.*, Dec. 2007, pp. 421–430

[30] M. Christodorescu, S. Jha, and C. Kruegel, "Mining specifications of malicious behavior," in *Proc. 6th Joint Meeting Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng.*, Sep. 2017, pp. 5–14.

[31] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, "Scalable, behavior-based malware clustering," in *Proc. NDSS*, 2009, pp. 8–11.

[32] I. Santos, J. Nieves, and P. G. Bringas "Semi-supervised learning for unknown malware detection," in *Proc. Int. Symp. Distrib. Comput. Artif. Intell.* Berlin, Germany: Springer, 2011, pp. 415–422.

[33] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, "Malware classification with deep convolutional neural networks," in *Proc. 9th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS)*, Feb. 2018, pp. 1–5.

[34] R. Kumar, Z. Xiaosong, R. U. Khan, I. Ahad, and J. Kumar, "Malicious code detection based on image processing using deep learning," in *Proc. Int. Conf. Comput. Artif. Intell.*, Mar. 2018, pp. 81–85.

[35] Z. Cui, X. Fei, X. Cai, C. Yang, G. G. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3187–3196, Jul. 2018.

[36] B. Xiaofang, C. Li, H. Weihua, and W. Qu, "Malware variant detection using similarity search over content fingerprint,"in *Proc. 26th Chin. Control Decis. Conf.*, May/Jun. 2014, pp. 5334–5339.

[37] J. Bouvriem, "Notes on convolutional neural networks," Tech. Rep., 2006.

[38] J. H. Paik, "A novel TF-IDF weighting scheme for effective ranking," in *Proc. 36th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2013, pp. 343–352.

[39] E. Haddi, X. Liu, and Y. Shi, "The role of text pre-processing in sentiment analysis," *Procedia Comput. Sci.*, vol. 17, pp. 26–32, Nov. 2013.

[40] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and M. Kudlur, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement.*, 2016, pp. 265–283.

[41] D. Baylor, "TFX: A tensorflow-based production-scale machine learning platform," in*Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 1387–1395.

[42] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks," 2014, *arXiv:1412.6830*. [Online]. Available: https://arxiv.org/abs/1412.6830

[43] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Netw.*, vol. 107, pp. 3–11, Nov. 2018.

[44] A. Tato and R. Nkambou, "Improving Adam optimizer," to be published.

[45] Z. Zhang, "Improved adam optimizer for deep neural networks," in *Proc. IEEE/ACM 26th Int. Symp. Qual. Service (IWQoS)*, Jun. 2018, pp. 1–2.

[46] U. Bandara and G. Wijayrathna, "Detection of source code plagiarism using machine learning approach," *Int. J. Comput. Theory Eng.*, vol. 4, no. 5, p. 674, 2012.

[47] F. Ullah, J. Wang, M. Farhan, S. Jabbar, Z. Wu, and S. Khalid, "Plagiarism detection in students' programming assignments based on semantics: Multimedia e-learning based smart assessment methodology," *Multimedia Tools Appl.*, pp. 1–18, 2018.

[48] Z. Cui, L. Du, P. Wang, X. Cai, and W. Zhang, "Malicious code detection based on CNNs and multi-objective algorithm," *J. Parallel Distrib. Comput.*, vol. 129, pp. 50–58, Jul. 2019.

FARHAN ULLAH received the B.S. degree in computer science from the University of Peshawar, Pakistan, in 2008, and the M.S. degree in computer science from CECOS University Peshawar, Pakistan, in 2012. He is currently pursuing the Ph.D. degree in computer science with the School of Computer Science, Sichuan University, Chengdu, China. He is a Lecturer at COMSATS University Islamabad, Sahiwal. His research work is published in various renowned journals of Springer, Elsevier, Wiley, MDPI, and Hindawi. His research interests include software similarity, information security, and data science. He received Research Productivity Award from the COMSATS Institute of Information Technology (CIIT), Sahiwal, Pakistan, in 2016.

HAMAD NAEEM received the Ph.D. degree in software engineering from the College of Computer Science, Sichuan University, China, in 2019. He has published various articles in reputed SCIE and EI journals/conferences. His research interests include malware detection, image processing, internet security, and machine learning.

SOHAIL JABBAR was a Postdoctoral Researcher with Kyungpook National University, Daegu, South Korea. He also served as an Assistant Professor with the Department of Computer Science, COMSATS Institute of Information Technology (CIIT), Sahiwal, and also headed the Networks and Communication Research Group, CIIT. He is currently an Assistant Professor with the Department of Computer Science, and the Director of Graduate Programs with the Faculty of Sciences, National Textile University, Faisalabad, Pakistan. He is also Head of the Network Communication and Media Analytics Research Group, National Textile University. He has authored one book, two book chapters, and over 70 research papers. His research work is published in various renowned journals and magazines of the IEEE, Springer, Elsevier, MDPI, Old City Publication, and Hindawi, and in conference proceedings of the IEEE, ACM, and IAENG. He has engaged in collaborative research with renowned research centers and institutes around the globe on various issues in the domains of the Internet of Things, wireless sensor networks, and big data. He has received many awards and honors from the Higher Education Commission of Pakistan, Bahria University, CIIT, and the Korean Government. Among those awards are the Best Student Research Awards of the Year, the Research Productivity Award, and the BK-21 Plus Postdoctoral Fellowship. He received the Research Productivity Award from CIIT, in 2014 and 2015. He has been involved in many national and international level projects. He has been a Reviewer for leading journals, including the ACM TOSN, JoS, MTAP, AHSWN, and ATECS, and conferences such as C-CODE 2017, ACM SAC 2016, and ICACT 2016. He is currently a TPC member/Chair in many conferences. He is a Guest Editor of the *Sis in Concurrency and Computation Practice and Experience* (Wiley), *Future Generation Computer Systems* (Elsevier), *Peer-to-Peer Networking and Applications* (Springer), the *Journal of Information and Processing System* (KIPS), *Cyber Physical System* (Taylor & Francis), and the IEEE WIRELESS COMMUNICATIONS (IEEE Communication Society).

SHEHZAD KHALID received the Bachelor of Science degree in computer systems engineering from the GIK Institute of Engineering Sciences and Technology, Topi, in 2000, and the master's degree in M.Sc. software engineering from the National University of Science and Technology, Rawalpindi, in 2003. And awarded to the Ph.D. degree in motion data mining and machine learning, February, from the University of Manchester, U.K., in 2009. He is currently a Professor with the Department of Computer Engineering, Bahria University, Islamabad. He is also designated as a Director (ORIC) in Head Office of Bahria University. His research areas include computer vision and pattern recognition. He is also headed by research group "Computer Vision and Pattern Recognition Research Center." He has completed his 76 publications so far including eight ISI indexed and 68 journal articles and 43 conference papers. He has completed nine research project and published five books chapters and one complete book. He was a recipient of 18 different awards, which is including a Best Researcher and a Best Teacher from Bahria University and Higher Education Commission Pakistan and other organizations. He is also supervised 17 M.S. students and eight Ph.D. students.

MUHAMMAD AHSAN LATIF received the Ph.D. degree in computer science from Alpen-Adria University, Klagenfurt, Austria, in 2011. He is currently serving as an Assistant Professor with the Department of Computer Science, University of Agriculture, Faisalabad, Pakistan. His main research interests include modeling, image processing, computer vision, and artificial intelligence in the agricultural context. Among his contributions are research articles, book chapters, a complete book on digital image processing, and various research projects. At present, he is leading a research group aiming to map crop health under different nonlinear conditions using unmanned aerial systems and vision sensors.

FADI AL-TURJMAN (M'07) received the Ph.D. degree in computer science from Queen's University, Kingston, ON, Canada, in 2011. He is currently a Professor with the Department of Computer Engineering, Antalya Bilim University, Antalya, Turkey. He is a leading authority in the areas of smart/cognitive, wireless, and mobile networks' architectures, protocols, deployments, and performance evaluation. His publication history spans over 200 publications in journals, conferences, patents, books, and book chapters, in addition to numerous keynotes and plenary talks at flagship venues. He has authored or edited more than 20 books about cognition, security, and wireless sensor networks' deployments in smart environments, published by Taylor & Francis and Springer. He has received several recognitions and Best Paper Awards at top international conferences. He also received the prestigious Best Research Paper Award from the *Computer Communications Journal* (Elsevier), from 2015 to 2018, in addition to the Top Researcher Award for 2018 at Antalya Bilim University. He has led a number of international symposia and workshops in flagship communication society conferences. He currently serves as the Lead Guest Editor for several well-reputed journals, including the *Elsevier Computer Communications* (COMCOM), the *Sustainable Cities & Society* (SCS), the *IET Wireless Sensor Systems*, and the Springer EURASIP and MONET journals.

LEONARDO MOSTARDA received the Ph.D. degree from the Computer Science Department of University of L'Aquila, in 2006. In 2007, he was a Research Associate with the Computing Department, Distributed Systemand Policy Group, Imperial College London. There he was working on the UBIVAL EPRC Project in cooperation with Cambridge, Oxford, Birmingham and UCL for building a novel middleware to support the programming of body sensor networks. In 2010, he was a Senior Lecturer with Middlesex University in the Distributed Systems and Networking Department. There he founded the Senso. Lab an Innovative Research Laboratory for building energy efficient wireless sensor networks. He is currently an Associate Professor and the Head of the Computer Science Department, Camerino University, Italy. Afterward, he cooperated with the European Space Agency (ESA) on the CUSPIS FP6 project to design and implement novel security protocols and secure geo tags for works of art authentication. To this end, he was combining traditional security mechanisms and satellite data.